PCT/JP2004/006970

日本国特許庁 01.7.2004
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年 ６月 ６日

出願番号
Application Number: 特願２００３－１６２９２８
[ST. 10/C]: [ＪＰ２００３－１６２９２８]

出願人
Applicant(s): シャープ株式会社

２００４年 ５月１２日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号 出証特２００４－３０３９５７７

【書類名】　特許願
【整理番号】　03J02117
【提出日】　平成15年 6月 6日
【あて先】　特許庁長官　殿
【国際特許分類】　G11B 27/036
【発明者】
　【住所又は居所】　大阪府大阪市阿倍野区長池町２２番２２号　シャープ株式会社内
　【氏名】　木田　歩
【発明者】
　【住所又は居所】　大阪府大阪市阿倍野区長池町２２番２２号　シャープ株式会社内
　【氏名】　木山　次郎
【発明者】
　【住所又は居所】　大阪府大阪市阿倍野区長池町２２番２２号　シャープ株式会社内
　【氏名】　山口　孝好
【特許出願人】
　【識別番号】　000005049
　【氏名又は名称】　シャープ株式会社
【代理人】
　【識別番号】　100080034
　【弁理士】
　【氏名又は名称】　原　謙三
　【電話番号】　06-6351-4384
【選任した代理人】
　【識別番号】　100113701
　【弁理士】
　【氏名又は名称】　木島　隆一

【選任した代理人】

【識別番号】　100116241

【弁理士】

【氏名又は名称】　金子　一郎

【手数料の表示】

【予納台帳番号】　003229

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0208489

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　データ記録方法、データ記録装置、およびデータ記録媒体

【特許請求の範囲】

【請求項１】

記録媒体に、ＡＶデータと、該ＡＶデータと同期するアフレコデータとを記録するデータ記録方法であって、

記録媒体上に、ＡＶデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能であり、

ユーザプログラムとアフレコデータとの関係を管理する管理情報を上記記録媒体上に記録することを特徴とするデータ記録方法。

【請求項２】

上記アフレコデータは、複数のユーザプログラムにおいて参照されるアフレコデータ群を含む領域中の部分領域であり、前記領域はひとつのアフレコファイルとして記録されており、

上記管理情報は、オリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコデータ毎に記録されることを特徴とする前記請求項１に記載のデータ記録方法。

【請求項３】

上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、

上記管理情報は、複数のオリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコファイル毎に記録されることを特徴とする前記請求項１に記載のデータ記録方法。

【請求項４】

上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、

上記管理情報は、オリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコファイル毎に記録されることを特徴とする前記請求項１に記載の

データ記録方法。

【請求項５】

上記管理情報は、アフレコデータが複数のユーザプログラムから参照されている場合における参照回数を含むことを特徴とする前記請求項１に記載のデータ記録方法。

【請求項６】

上記管理情報は、アフレコデータを参照しているユーザプログラム名を含むことを特徴とする前記請求項１に記載のデータ記録方法。

【請求項７】

上記管理情報は、ユーザプログラムによって参照されているアフレコデータの参照範囲の情報を含むことを特徴とする前記請求項１に記載のデータ記録方法。

【請求項８】

記録媒体に、ＡＶデータと、該ＡＶデータと同期するアフレコデータとを記録するデータ記録装置であって、

記録媒体上に、ＡＶデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能であり、

ユーザプログラムとアフレコデータとの参照関係を管理する管理情報を上記記録媒体上に記録することを特徴とするデータ記録装置。

【請求項９】

ＡＶデータと、該ＡＶデータと同期するアフレコデータとが記録されたデータ記録媒体であって、

記録媒体上に記録したＡＶデータの同一のシーンを使って作成された複数のユーザプログラムが、複数のユーザプログラムによって同一のアフレコデータを参照可能とするように記録されており、

さらに、ユーザプログラムとアフレコデータとの参照関係を管理する管理情報が記録されていることを特徴とするデータ記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、映像データ、音声データをハードディスク、光ディスク等のランダムアクセス可能な記録媒体に対して記録・再生するデータ記録方法及びデータ記録装置に関するものである。

【０００２】

【従来の技術】

ディスクメディアや半導体媒体を用いたビデオや音声のディジタル記録再生装置が普及しつつある。それらにおいて、テープメディアと同様アフターレコーディング（アフレコ）機能を安価に実現する技術が求められている。アフレコ機能は、既に記録したオーディオやビデオに対し、後から情報、特にオーディオやグラフィックを追記する機能である。

【０００３】

ここで、テープメディアにはないディスクメディアにおける特徴機能である、非破壊編集機能あるいはノンリニア編集機能と呼ばれる機能がある。すなわち、非破壊編集機能あるいはノンリニア編集機能では、ディスク上に記録したＡＶストリームを移動あるいはコピーすることなく、ＡＶストリームの任意の区間を任意の順番で再生できる。この機能においては、単に再生順番を変えるだけではなく、同一のシーンを使って複数のユーザプログラムを作る、つまり同一のシーンを複数のユーザプログラムで共有することが可能となる。このときのユーザの要望としては、それぞれのユーザプログラムで個別にアフレコがしたい、というものがある。

【０００４】

例えば、ユーザプログラムＡではあるシーンに対してＢＧＭをアフレコし、同じシーンを共有するユーザプログラムＢではアナウンスをアフレコしたい、といった要望である。そのような要望に対して、特許文献１では、アフレコ用領域割り当てを管理する情報を備えることで、複数のユーザプログラムからアフレコした場合に、異なるユーザプログラム間で共有される同一のシーンのアフレコに対して、意図しない上書きをすることを防ぐことができる。

【０００５】

上記特許文献１におけるディスクの記録フォーマットを図２７に示す。ディスクにおける記録領域は、ＥＣＣ（Error Correcting Code）ブロックの列で構成される。ＥＣＣブロックは符号化を行う際の最小単位であり、データに加えエラー補正用のパリティが付加され、符号化が行われている。ディスク内のデータを読み込む際は、このＥＣＣブロックの単位で読み込み誤り訂正をしてから、必要なデータを取り出す。

【０００６】

一方、ディスクに対してデータの書き換えを行う際は、まずＥＣＣブロック単位で読み込み、誤り訂正をしたデータに対して必要な部分を書き換え、再度誤り符号を付与し、ディスクに記録を行う。

【０００７】

ビデオ（映像）データやオーディオ（音声）データは、ＥＣＣブロック中で、アフレコデータブロック、オリジナルオーディオブロック、オリジナルビデオブロックの順に配置される。それぞれのブロックにはほぼ同じ時間に対応するアフレコデータ、オリジナルオーディオデータ、オリジナルビデオデータが含まれている。

【０００８】

尚、オリジナルオーディオブロックとオリジナルビデオブロックとを合わせてオリジナルブロックと呼ぶことにする。オリジナルプログラム（アフレコデータを記録する前の映像）を記録する際は、アフレコデータブロックにダミーのデータを書き込んでおく。

【０００９】

アフレコデータの記録の際は、アフレコデータブロックへデータの書込みを行う。このとき、アフレコデータブロック中の記録を行った領域については既割り当て領域とし、残りを未割り当て領域としてオリジナルプログラム中の管理情報内で別々に管理する。この管理情報を参照することで、異なるユーザプログラム間でのアフレコデータの上書きを防止することができる。新たなアフレコデータが記録された際には、それぞれの領域における管理情報の書き換えを行う。

【００１０】

【特許文献１】

特開２００２－３７３４８０号公報（公開日平成１４年１２月２６日）

【００１１】

【発明が解決しようとする課題】

ところが、上記従来の構成では、ユーザプログラムを削除する場合を考えると、削除されるユーザプログラムに対応して不要となったアフレコデータを削除して領域を開放し、アフレコデータブロック領域を有効に再利用したいという要求に対しては、以下のような問題が生じる。

【００１２】

すなわち、同一のシーンを複数のユーザプログラムで共有する場合、各アフレコデータは複数のユーザプログラムから参照されている可能性があり、これを安易に削除することはできない。これを、より具体的に説明すると以下のとおりである。

【００１３】

つまり、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータを併せて削除しようとする場合には、削除されるべきアフレコデータは削除されるユーザプログラムのみで参照されるものであり、他のユーザプログラムにおいて共有されていないことが必要である。

【００１４】

しかしながら、従来では、ユーザプログラムの削除を行った場合、参照していたアフレコデータが他のどのユーザプログラムからも参照されておらず削除可能であると判断されるためには、すべてのユーザプログラムの管理情報をチェックする必要が生じる。このようなチェック動作は、シーク時間の長い光ディスクにおいては処理時間の面で問題となる。

【００１５】

本発明は、上記の問題点を解決するためになされたもので、その目的は、ユーザプログラムの削除時、該ユーザプログラムにおいて参照していたアフレコデータが削除可能であるか否かの判断を容易にすることができるデータ記録方法およ

びデータ記録装置を提供することにある。

【００１６】

【課題を解決するための手段】

本発明のデータ記録方法は、上記の課題を解決するために、記録媒体に、ＡＶデータと、該ＡＶデータと同期するアフレコデータとを記録するデータ記録方法であって、記録媒体上に、ＡＶデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能であり、ユーザプログラムとアフレコデータとの関係を管理する管理情報を上記記録媒体上に記録することを特徴としている。

【００１７】

上記の構成によれば、記録媒体に、ＡＶデータと、該ＡＶデータと同期するアフレコデータと記録するにあたって、記録媒体上に、ＡＶデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能となっている。

【００１８】

このため、例えば、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータを併せて削除しようとする場合には、削除されるべきアフレコデータは削除されるユーザプログラムのみで参照されるものであることを確認することが必要となるが、このような確認が上記管理情報によって容易に行える。

【００１９】

また、上記データ記録方法では、上記アフレコデータは、複数のユーザプログラムにおいて参照されるアフレコデータ群を含む領域中の部分領域であり、前記領域はひとつのアフレコファイルとして記録されており、上記管理情報は、オリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコデータ毎に記録される構成とすることができる。

【００２０】

また、上記データ記録方法では、上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録さ

れており、上記管理情報は、複数のオリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコファイル毎に記録される構成とすることができる。

【００２１】

上記の構成によれば、アフレコファイルのサイズとアドレスとが常に変わらないような管理が可能であり、アフレコデータの削除が管理情報の書き換えのみで行うことができるため、ファイルシステムの書き換えを必要としない。

【００２２】

また、上記データ記録方法では、上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、上記管理情報は、オリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコファイル毎に記録される構成とすることができる。

【００２３】

上記の構成によれば、アフレコデータ毎にアフレコファイルが分かれているため、アフレコデータを特定する際、データのオフセットなどから検索しなくとも、各アフレコファイルの参照ファイル名から即座に特定できるため、データの特定がより簡便に、より迅速に行うことができる。

【００２４】

また、上記データ記録方法では、上記管理情報は、アフレコデータが複数のユーザプログラムから参照されている場合における参照回数を含む構成とすることができる。

【００２５】

上記の構成によれば、例えば、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータにおいて、該アフレコデータがユーザプログラムから参照されている参照回数が1減らされる。そして、この参照回数を減らす処理によって参照回数が０となったアフレコデータは、削除されるユーザプログラムのみで参照されるものであることが容易に確認できる。

【００２６】

また、上記データ記録方法では、上記管理情報は、アフレコデータを参照しているユーザプログラム名を含む構成とすることができる。

【００２７】

上記の構成によれば、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータの削除が指示された場合に、該アフレコデータが他のユーザプログラムファイルから参照されているか否かの情報の他に、どのファイルから参照されているかをユーザに提示することができる。

【００２８】

また、上記データ記録方法では、上記管理情報は、ユーザプログラムによって参照されているアフレコデータの参照範囲の情報を含む構成とすることができる。

【００２９】

上記の構成によれば、各ユーザプログラムからアフレコデータのどの部分が参照されているかを示す参照範囲の情報を持たせることで、このような参照範囲情報により、アフレコデータ自体は複数のユーザプログラムから参照されていても要らなくなった部分のみを削除できるため、より有効に記録媒体の記録領域を活用することができる。

【００３０】

【発明の実施の形態】

以下、本発明の実施形態について、図面を参照しながら詳細に説明する。ここでの説明は、本発明において共通に用いる構成、個々の実施形態に固有の内容という順に行っていく。

【００３１】

＜システム構成＞

図２は、本実施の形態において共通に用いる、アフレコ可能なビデオディスクレコーダの構成である。

【００３２】

このビデオディスクレコーダは、図２に示すように、バス１００、ホストＣＰ

U（Central Processing Unit）１０１、ＲＡＭ（Random Access Memory）１０２、ＲＯＭ（Read Only Memory）１０３、ユーザインタフェース１０４、システムクロック１０５、光ディスク１０６、ピックアップ１０７、ＥＣＣ（Error Correcting Code）デコーダ１０８、ＥＣＣエンコーダ１０９、再生用バッファ１１０、記録／アフレコ用バッファ１１１、デマルチプレクサ１１２、マルチプレクサ１１３、多重化用バッファ１１４、オーディオデコーダ１１５、ビデオデコーダ１１６、オーディオエンコーダ１１７、ビデオエンコーダ１１８および図示されないカメラ、マイク、スピーカ、ディスプレイ等で構成される。

【００３３】

ホストＣＰＵ１０１は、バス１００を通じてデマルチプレクサ１１２、マルチプレクサ１１３、ピックアップ１０７、また図示していないが、オーディオデコーダ１１５、ビデオデコーダ１１６、オーディオエンコーダ１１７、ビデオエンコーダ１１８との通信を行う。

【００３４】

再生時には、ピックアップ１０７を通じて光ディスク１０６から読み出されたデータは、ＥＣＣデコーダ１０８によって誤り訂正され、再生用バッファ１１０に一旦蓄えられる。デマルチプレクサ１１２は、オーディオデコーダ１１５、ビデオデコーダ１１６からのデータ送信要求に従って、再生用バッファ１１０中のデータをその種別によって適切なデコーダに振り分ける。

【００３５】

一方、記録時には、オーディオエンコーダ１１７とビデオエンコーダ１１８によって圧縮符号化されたデータは、マルチプレクサ１１３を介して多重化用バッファ１１４に一旦送られ、さらに、マルチプレクサ１１３によってＡＶ多重化されて多重化用バッファ１１４から記録／アフレコ用バッファ１１１に送られる。記録／アフレコ用バッファ１１１中のデータは、ＥＣＣエンコーダ１０９によって誤り訂正符号を付加され、ピックアップ１０７を通じて光ディスク１０６に記録される。

【００３６】

オーディオデータの符号化方式にはMPEG-1 Layer-IIを、ビデオデータの符号

化方式にはMPEG-2をそれぞれ用いることができる。

【００３７】

光ディスク１０６は、外周から内周に向かって螺旋状に記録再生の行われるものであり、ビデオディスクレコーダに対して脱着可能な記録媒体とする。また、光ディスク１０６は、２０４８ｂｙｔｅを１セクタとし、誤り訂正のため１６セクタでＥＣＣブロックを構成する。ＥＣＣブロック中のデータを書き換える場合には、そのデータが含まれるＥＣＣブロック全体を読み込み、誤り訂正を行い、対象のデータを書き換え、再び誤り訂正符号を付加し、ＥＣＣブロックを構成して光ディスク１０６に記録する必要がある。また、光ディスク１０６では、記録効率を上げるためＺＣＡＶ（ゾーン角速度一定）を採用しており、記録領域は回転数の異なる複数のゾーンで構成される。

【００３８】

＜ファイルシステム＞

本実施の形態に係るビデオディスクレコーダでは、光ディスク１０６上の各種情報を管理するためにファイルシステムを用い、このファイルシステムをデータ（アフレコデータ、オーディオデータ、ビデオデータ）と共に光ディスク１０６に記録する。ファイルシステムには、パーソナルコンピュータ（以下、ＰＣと略称する）との相互運用を考慮してＵＤＦ（Universal Disk Format）を使用することができる。ファイルシステム上では、各種管理情報やＡＶストリームはファイルとして扱われる。

【００３９】

ファイルシステムのユーザエリアは、２０４８ｂｙｔｅの論理ブロック(セクタと一対一に対応)で管理される。ユーザエリア上の各ファイルは、整数個のエクステント(連続した論理ブロック)で構成され、エクステント単位で分散して記録してもよい。ユーザエリア内の空き領域は、Space Bitmapを用いて論理ブロック単位で管理される。

【００４０】

＜ファイルフォーマット＞

ＡＶストリーム管理のためのフォーマットとして、ここではQuickTimeファイ

ルフォーマットを用いる。QuickTimeファイルフォーマットとは、Apple社が開発したマルチメディアデータ管理用フォーマットであり、ＰＣの世界では広く用いられている。

【００４１】

QuickTimeファイルフォーマットは、ビデオデータやオーディオデータ等(これらを総称してメディアデータとも呼ぶ)と管理情報とで構成される。メディアデータと管理情報とを合わせてここでは、QuickTimeムービー(略してムービー)と呼ぶ。メディアデータと管理情報とは、同じファイル中に存在しても、別々のファイルに存在してもよい。

【００４２】

メディアデータと管理情報とが同じファイル中に存在する場合は、図３（ａ）に示すようなファイル構成をとる。各種情報はatomという共通の構造に格納される。ファイル２０１において、管理情報はMovie atomという構造に格納され、メディアデータのＡＶストリームはMovie data atomという構造に格納される。また、Movie data atomの最初には、Atom Header（ＡＨ）が配置される。尚、Movie atom中の管理情報には、メディアデータ中の任意の時間に対応するＡＶストリームのファイル中での相対位置を導くためのテーブルや、メディアデータの属性情報や、後述する外部参照情報等が含まれている。

【００４３】

一方、管理情報とメディアデータとを別々のファイルに格納した場合は、図３（ｂ）に示すような構成をとる。この場合、管理情報はファイル２０２においてMovie atomという構造に格納されるが、ＡＶストリームはatomには格納される必要はなく、ファイル２０３に格納される。このとき、Movie atomはＡＶストリームを格納したファイル２０３を「外部参照」している、という。

【００４４】

外部参照では、図３（ｃ）に示すように、管理情報を格納する一つのファイル２０４によって複数のＡＶストリームファイル（この図ではファイル２０５・２０６）を参照することが可能である。この仕組みにより、ＡＶストリーム自体を物理的に移動することなく、見かけ上編集を行ったように見せる、いわゆる「ノ

ンリニア編集」「非破壊編集」が可能になる。また、光ディスク１０６上に記録したＡＶストリームの同一のシーンを使って複数のユーザプログラムを作成することが可能であり、かつ、同一のアフレコデータを複数のユーザプログラムにて参照することも可能である。

【００４５】

次に、図４ないし図１４を用いて、QuickTimeの管理情報のフォーマットについて説明する。まず、共通の情報格納フォーマットであるatomについて説明する。atomの先頭には、そのatomのサイズであるAtom size、そのatomの種別情報であるTypeが必ず存在する。Typeは４文字で区別され、例えばMovie atomでは‘moov’、Movie data atomでは‘mdat’となっている。

【００４６】

atom間には階層構造があり、各atomは別のatomを含むことができる。Movie atomの構成を図４に示す。Movie header atom は、そのMovie atomが管理するムービーの全体的な属性を管理する。Track atomは、そのムービーに含まれるビデオやオーディオ等のトラックに関する情報を格納する。User data atomは、ユーザが独自に定義可能なatomである。

【００４７】

Track atomの構成を図５に示す。Track header atomはそのトラックの全体的な属性を管理する。Edit atomは、メディアデータのどの区間をムービーのどのタイミングで再生するかを管理する。Track reference atomは、別のトラックとの関係を管理する。Media atomは、実際のビデオやオーディオといったデータを管理する。

【００４８】

Track header atomの構成を図６に示す。ここでは、後での説明に必要なもののみ説明する。Flagsは属性を示すフラグの集合である。代表的なものとして、Track enabledフラグがあり、このフラグが‘１’であればそのトラックは再生され、‘０’であれば再生されない。Track IDはトラック固有のIDである。Layerはそのトラックの空間的な優先度を表しており、画像を表示するトラックが複数あれば、Layerの値が小さいトラックほど画像が前面に表示される。

【００４９】

Media atomの構成を図７に示す。Media header atomは、そのMedia atomの管理するメディアデータに関する全体的な属性等を管理する。Handler reference atomは、メディアデータをどのデコーダでデコードするかを示す情報を格納する。Media information atomは、ビデオやオーディオ等のメディア固有の属性情報を管理する。

【００５０】

Media information atomの構成を図８に示す。Media information header atomは、ビデオやオーディオ等のメディア固有の属性情報を管理する。Handler reference atomは、Media atomの項で説明した通りである。Data information atomは、そのQuickTimeムービーが参照するメディアデータを含むファイルの名前を管理するatomであるData reference atomを含む。Sample table atomは、データのサイズや再生時間等を管理している。

【００５１】

次にSample table atomについて説明するが、その前に、QuickTimeにおけるデータの管理方法について、図９を用いて説明する。QuickTimeでは、データの最小単位(例えばビデオフレーム)をサンプル（sample）と呼ぶ。個々のトラック毎に、サンプルには再生時間順に１から番号(サンプル番号)がついている。

【００５２】

また、QuickTimeフォーマットでは、個々のサンプルの再生時間長およびデータサイズを管理している。また、同一トラックに属するサンプルが再生時間順にファイル中で連続的に配置された領域をチャンク（chunk）と呼ぶ。チャンクにも再生時間順に、１から番号がついている。すなわち、QuickTimeフォーマットでは、このようにチャンクが連続して配置されることにより一つのファイル９０１が形成される。

【００５３】

さらに、QuickTimeフォーマットでは、個々のチャンクにおけるファイル先頭からのアドレスおよび個々のチャンクが含むサンプル数を管理している。これらの情報に基づき、任意の時間に対応するサンプルの位置を求めることが可能とな

っている。

【００５４】

Sample table atomの構成を図１０に示す。Sample description atomは、個々のチャンクのデータフォーマット(Data format)やサンプルが格納されているファイルのチャンクのインデックス等を管理する。さらに、Sample description atomの構成を図１１（ａ）に示す。number-of-entriesは使用されているデータフォーマットの数を表し、その数だけSample description tableが存在する。図１１（ｂ）はオーディオのSample description tableの例である。

【００５５】

図１０のSample table atomにおいて、Time-to-sample atomは、個々のサンプルの再生時間を管理する。Sync sample atomは、個々のサンプルのうち、デコード開始可能なサンプルを管理する。Sample-to-chunk atomは、個々のチャンクに含まれるサンプル数を管理する。Sample size atomは、個々のサンプルのサイズを管理する。Chunk offset atomは、個々のチャンクのファイル先頭からのアドレスを管理する。

【００５６】

Edit atomは、図１２に示すように１個のEdit list atomを含む。Edit list atomはNumber of entriesで指定される個数分の、Track duration、Media time、Media rateの値の組(エントリ)を持つ。各エントリは、トラック上で連続的に再生される区間に対応し、そのトラック上での再生時間順に並んでいる。

【００５７】

Track durationは、そのエントリが管理する区間のトラック上での再生時間、Media timeは、そのエントリが管理する区間の先頭に対応するメディアデータ上での位置、Media rateは、そのエントリが管理する区間の再生スピードを表す。尚、Media timeが‘－１’の場合は、そのエントリのTrack duration分、そのトラックでのサンプルの再生を停止する。この区間のことをempty editと呼ぶ。

【００５８】

図１３にEdit listの使用例を示す。ここでは、Edit list atomの内容が図１３（ａ）に示す内容であり、さらにサンプルの構成が図１３（ｂ）に示す構成で

あるとする。尚、ここではｉ番目のエントリのTrack durationをD(i)、Media timeをT(i)、Media rateをR(i)として示している。このとき、実際のサンプルの再生は、図１３（ｃ）に示す順に行われる。このことについて簡単に説明する。

【００５９】

まず、エントリ#1は、Track durationが13000、Media timeが20000、Media rateが１であるため、そのトラックの先頭から13000の区間はサンプル中の時刻20000から33000の区間を再生する。次に、エントリ#2は、Track durationが5000、Media timeが－１であるため、トラック中の時刻13000から18000の区間、何も再生を行わない。そして、エントリ#3は、Track durationが10000、Media timeが0、Media rateが１であるため、トラック中の時刻18000から28000の区間において、サンプル中の時刻0から10000の区間を再生する。

【００６０】

図１４に、User data atomの構成を示す。このatomには、QuickTimeフォーマットで定義されてない独自の情報を任意個数格納することができる。１個の独自情報は１個のエントリで管理され、１個のエントリはAtom sizeとTypeとUser dataとで構成される。Atom sizeはそのエントリ自体のサイズを表し、Typeは独自情報をそれぞれ区別するための識別情報、User dataは実際のデータを表す。

【００６１】

＜インデックスファイル＞

ディスク内に含まれるQuickTimeムービーを管理するため、インデックスファイルという特別なQuickTimeムービーファイルをディスク内に１個置く。

【００６２】

インデックスファイルには、ディスク内のファイル(QuickTimeムービーやQuickTimeムービーから参照されている静止画等)に関するサムネイルや各種属性が登録されている。各種属性の中には、そのファイルが外部参照されている回数を示すlink countがある。

【００６３】

link countを参照することで、そのファイルを参照しているファイルがあるかどうかを容易に知ることができ、他から参照されているファイルを不用意に削除

してしまうことを防ぐことができる。

【００６４】

＜ＡＶストリームの形態＞

ＡＶストリームの形態を図１５に示す。ＡＶストリームは整数個のContinuous Unit（ＣＵ）で構成される。ＣＵはディスク上で連続的に記録される単位である。ＣＵの境界はＥＣＣブロック境界と一致するようにストリームが構成されている。またＣＵは、一つのPost Record Unit（ＰＲＵ）と整数個のVideo Unit（ＶＵ）とで構成される。ＶＵは独立に再生可能な単位であり、ＶＵの集合が前述のオリジナルブロックに対応する。ＰＲＵは前述のアフレコデータブロックに対応する。ＰＲＵの領域サイズは、ＣＵの再生時間にオーディオの最大ビットレートをかけたものである。

【００６５】

＜ファイルシステムフォーマット＞

本発明の説明において用いるファイルシステムのフォーマットである、ＵＤＦ（Universal Disk Format）について図１６を用いて説明する。図１６(ａ)に示すディレクトリ／ファイル構成をＵＤＦで記録した例を図１６(ｂ)に示す。図中のＡＶＤＰ６０２はAnchor Volume Descriptor Pointerの略であり、ＵＤＦの管理情報を探すためのエントリポイントに相当し、通常２５６セクタ目、Ｎセクタ目、あるいはＮ－２５６セクタ目（Ｎは最大論理セクタ番号）に記録される。ＶＤＳ６０１はVolume Descriptor Sequenceの略であり、ＵＤＦが管理する領域であるボリュームに関する管理情報を記録する。ボリュームは一般に一枚のディスクに１個存在し、その中にパーティションを一般に１個含む。ＦＳＤ６０３はFile Set Descriptorの略であり、パーティションに１個存在する。パーティションの中での位置情報はパーティションの先頭からのセクタ番号に相当する論理ブロック番号で示される。なお、１個の論理ブロックは１セクタに対応する。また、各パーティションには図示しないがSpace Bitmapと呼ばれる各論理ブロックがファイルにすでに割り当てられているかそうでないかを示すテーブルが存在する。

【００６６】

ＦＳＤ６０３は、ルートディレクトリのFile Entry（ＦＥ）であるＦＥ６０４の位置情報（論理ブロック番号と論理ブロック数で構成されextentと呼ばれる）を含む。ＦＥは、extentの集合を管理しており、extentを書き換えたり、追加したり、削除することで、ファイルを構成する実データの順番を変えたり、データを挿入したり削除したりすることが可能である。ＦＥ６０４はルートディレクトリの直下のファイルやディレクトリの名称等を格納するFile Identifier Descriptor（ＦＩＤ）の集合を格納する領域６０５を管理する。領域６０５中のＦＩＤ６１１、ＦＩＤ６１２は、それぞれファイル６２１、ファイル６２２のファイル名やextentの集合を管理するＦＥ６０６、ＦＥ６０８の位置情報を含む。ＦＥ６０６はファイル６２１の実データを構成する領域である領域６０７、領域６１０をextentとして管理する。ＦＥ６０８はファイル６２２の実データを構成する領域である領域６０９をextentとして管理する。このときファイル６２１の実データにアクセスするためには、ＡＶＤＰ６０２、ＶＤＳ６０１、ＦＳＤ６０３、ＦＥ６０４、ＦＩＤ６１１、ＦＥ６０６、領域６０７、領域６１０の順にリンクを辿っていけばよい。

【００６７】

〔実施の形態１〕

本発明の実施の形態１について、図１、図１７ないし図２４を用いて説明する。

【００６８】

ここでは、アフレコに関する処理を中心に説明し、オリジナルプログラムのＡＶストリームの処理については特に説明しない。

【００６９】

＜初期記録時の処理＞

ＰＲＵはオリジナルプログラムの記録の際にアフレコデータ用として確保される領域であり、ＡＶストリームの記録時にアフレコファイルが作成されて、この領域はダミーデータとして管理される。

【００７０】

オリジナルプログラムの管理情報内には、ＰＲＵ領域内のアフレコデータが記

録されていない領域を管理するための未割り当てトラックを作成し、初期記録の際は、ＣＵ単位のＰＲＵ領域を１サンプルとしてすべてのＰＲＵ領域についての位置、サイズ、時間情報を管理するsample tableを作成する。この未割り当てトラックについては、Track header atomのTrack Enabled フラグを０にセットし、再生対象ではないことを示す。

【００７１】

＜アフレコデータ記録時の処理＞

次に、オリジナルプログラムを外部参照したユーザプログラムを作成し、それにアフレコデータを記録する際の手順を説明する。ここでは、作成するユーザプログラムが、オリジナルプログラムの一部を抜き出して再生するというものである場合を考える。

【００７２】

ユーザプログラムの作成は、オリジナルプログラムの管理情報からＡＶストリームのトラックのユーザが指定した区間のsample tableをコピーすればよい。

【００７３】

このユーザプログラムにアフレコデータを追加するときの処理について図１に基づいて説明する。まず、ユーザプログラムのQuickTime管理情報をＲＡＭ１０２に読み込む。また、ユーザプログラムの管理情報のData reference atomに記述されたファイル名からオリジナルプログラムを特定し、参照先のオリジナルプログラムのQuickTime管理情報もＲＡＭ１０２に読み込む（Ｓ１５０１）。

【００７４】

次に、ユーザが指定した区間におけるアフレコデータを記録すべきＰＲＵの位置を、オリジナルプログラムの未割り当てトラックのsample tableから、アフレコ対象のビデオと時間的に対応する部分を探すことで特定する。そして、これから記録しようとするアフレコがオーディオの場合、ビットレートからデータ量を計算し、未割り当てトラックで管理しているＰＲＵのsize内に収まるかどうかを判定する（Ｓ１５０２）。

【００７５】

また、アフレコがグラフィックの場合は、記録するグラフィックデータのサイ

ズがＰＲＵのsize内に収まるかどうかを判定する。具体的には、アフレコデータは、ＣＵ単位のストリーム再生時間分ずつ、ディスク上に均等に配置されているＰＲＵ領域に分けて記録されるため、ユーザが指定したアフレコ区間全てのＰＲＵ未割り当て領域のsizeを調べて判定を行う。

【００７６】

アフレコデータを記録する領域が決定すれば、実際にその領域にアフレコデータを記録する。

【００７７】

アフレコデータの記録が終了すると、次に、オリジナルプログラム側にアフレコしたデータを管理するトラックを新規に作成し（Ｓ１５０３）、記録したアフレコデータを管理するsample tableを作成する（Ｓ１５０４）。

【００７８】

ここで、データの参照回数を管理するため、Track atom中のUser data atom('udta')内に、図１７に示すような、独自のAtom typeであるreference management atom('rfmg')を定義する。このreference management atomの中に、データが参照されている回数を示すフィールド‘ref#counter’を定義する。

【００７９】

すなわち、Sample tableを作成した後、User data atom内に上記reference management atomを作成し、ref#counterフィールドに参照回数が１回であることを表す‘1’をセットする（Ｓ１５０５）。

【００８０】

次に、オリジナルプログラムの未割り当てトラックのsample tableのchunk offsetとsizeを、アフレコデータ記録後のＰＲＵの残り領域に対応するよう更新する（Ｓ１５０６）。

【００８１】

次に、記録したアフレコデータを再生するためのトラックをユーザプログラム側にも新規に作成し、先程作成したオリジナルプログラムと同様、アフレコトラックのsample tableを作成する。

【００８２】

最後に、ＲＡＭ１０２上のオリジナルプログラムおよびユーザプログラムのQuickTime管理情報を光ディスク１０６に記録する（Ｓ１５０７）。

【００８３】

上記フローにおいて、Ｓ１５０２～Ｓ１５０６の処理は、図２に示す構成のビデオディスクレコーダでは、ホストＣＰＵ１０１が、ＲＡＭ１０２内に読み込まれたオリジナルプログラムおよびユーザプログラムのQuickTime管理情報に対して行う。そして、Ｓ１５０２～Ｓ１５０６の処理が施されたオリジナルプログラムおよびユーザプログラムのQuickTime管理情報が、ホストＣＰＵ１０１に制御によってＲＡＭ１０２から読み出されバス１００およびピックアップ１０７を介して光ディスク１０６に書き込まれる。

【００８４】

一方、Ｓ１５０２で未割り当て領域がなかった場合、アフレコ用のビットレートを下げることで割り当てる領域のサイズを小さくし、割り当て領域があるかどうかを再度調べるようにすることができる。それでも未割り当て領域がなければ、ＡＶストリームのＰＲＵ領域とは別の領域に記録を行うことも可能である。アフレコする区間について、部分的に記録する領域がない場合、その区間のみＰＲＵ領域とは別の領域に記録するようにしても良い。

【００８５】

アフレコデータ記録後のオリジナルプログラムは図１８のようになる。アフレコが記録された区間のＰＲＵは、アフレコデータが記録された領域(既割り当て領域)と未割り当て領域で構成される。オリジナルプログラムの管理情報であるMovie atomにはVideoトラック、Main Audioトラック、アフレコトラック(図中PR Audio)、未割り当てトラックの４つのトラックがあり、それぞれのデータのアドレスやサイズ、再生時間等の情報を持つ。

【００８６】

既にアフレコデータが記録されている後に別の種類のアフレコデータを追加するときの処理も、新規にファイルを作成しないこと以外は同様である。図１９（ａ）はアフレコデータ追加前のＰＲＵの状態であり、図１９（ｂ）はアフレコデータ追加後のＰＲＵの状態を示す。図１９（ｂ）では、既に記録されているアフ

レコ#Aトラックの後ろに、追加されるアフレコデータ用に新規にアフレコ#Bトラックを作成し、アフレコデータ毎に１トラックで管理を行う。Chunk offsetの値はＰＲＵ中の新規アフレコデータ開始位置をセットする。図では既にあったアフレコデータのすぐ後に新しいアフレコデータを記録しているが、未割り当て領域のどの位置でも記録を行って良い。新規データが追加される度に、残り領域に応じて未割り当てトラックのChunk offsetおよびSample sizeの更新を行う。

【００８７】

また、ファイルシステム上では図２０に示すように、オリジナルブロックで１ファイル（ストリームファイル）、アフレコデータブロックで１ファイル（アフレコファイル）の外部参照ファイルとして扱う。すなわち、アフレコファイル中に複数のアフレコデータが混在する形になる。

【００８８】

アフレコファイル中に複数のアフレコデータが存在している場合でも、アフレコデータの種類毎にトラックを分け、トラック毎にref#counterを持つことで、容易に参照回数を管理することができる。

【００８９】

＜ユーザプログラム作成時の処理＞

次に、アフレコデータを持つユーザプログラムをさらに非破壊編集する場合の処理を説明する。例えば図２１のようにユーザプログラム１とユーザプログラム２をつなぎ合わせたユーザプログラム３を作成するときを考える。

【００９０】

まず、ユーザプログラム１および２の管理情報をＲＡＭ１０２に読み込み、新たなユーザプログラム３として結合する。また、このユーザプログラム３のData reference atomに記述された参照ファイル名からオリジナルプログラムを特定し、オリジナルプログラムに関する管理情報をＲＡＭ１０２上に読み込む。

【００９１】

次に、オリジナルプログラムの管理情報に関して、ユーザプログラム３のアフレコトラックのchunk offset情報から、offsetが同一のアフレコ済みサンプルをサーチすることで、複数あるアフレコデータの中からユーザプログラム３が参照

しているアフレコデータを特定し、それに対応するtrack atom内のudta atomのreference management atom ('rfmg')にあるred#counterを１増やす。

【００９２】

最後に、ＲＡＭ１０２上のオリジナルプログラムおよびユーザプログラム３の管理情報を光ディスク１０６に記録する。

【００９３】

＜ユーザプログラム削除時の処理＞

ユーザから、ユーザプログラムの削除が指示された場合の処理について図２２に沿って説明する。尚、はじめに削除指示されたユーザプログラムの管理情報をＲＡＭ１０２上に読み込んでおく。

【００９４】

まず、ユーザプログラムの管理情報中のトラックを調べ、そのユーザプログラムが参照しているアフレコデータを管理しているオリジナルプログラムをリストアップする（Ｓ２１０１）。アフレコデータを管理するトラックがない場合は処理を終了する。

【００９５】

次に、リストアップした各オリジナルプログラムに関して、以下の処理を行う。まず、オリジナルプログラムの管理情報をＲＡＭ１０２に読み込む。そして、ユーザプログラムのアフレコトラックのchunk offset情報からオリジナルプログラム内の対応するアフレコトラックを検索し、検索されたトラックのref#counterを１減らす（Ｓ２１０２）。

【００９６】

ここでref#counterが０かどうかの判定を行い（Ｓ２１０３）、０であれば、削除を行ってこの部分の領域を開放し（Ｓ２１０４）、開放した領域を未割り当て領域に返却するように、未割り当てトラックのサンプル管理情報の更新を行う（Ｓ２１０５）。また、オリジナルプログラムの管理情報から削除を行ったアフレコデータに対応するTrack atomを削除する（Ｓ２１０６）。この処理はアフレコデータ記録時の処理の逆に相当する。

【００９７】

最後に、ユーザプログラムを格納しているファイルを光ディスク１０６から削除し（Ｓ２１０７）、オリジナルプログラムの管理情報を光ディスク１０６に記録する（Ｓ２１０８）。

【００９８】

また、Ｓ２１０３において、ref#counterが０でない場合には、上記アフレコデータを参照している他のユーザプログラムを存在していることを示しているので、Ｓ２１０４～２１０６の処理、すなわちアフレコデートの削除処理を行わずにＳ２１０７に移行する。

【００９９】

例えば、図２３に示すような参照関係を持つファイルがあり、このうちユーザプログラム３を削除する場合を考える。ここでユーザプログラム３においては、アフレコデータＡとアフレコデータＢとが参照されている。ユーザプログラム３を削除するにあたって、アフレコデータＡおよびアフレコデータＢのref#counter数を１減らすと、図２４に示すように、アフレコデータＡのref#counter数は１になり、アフレコデータＢのref#counter数は０になる。

【０１００】

すなわち、アフレコデータＢはユーザプログラム３にのみ参照されていたデータであるため、ユーザプログラム３と同時にアフレコデータＢも削除されるが、アフレコデータＡは他のユーザプログラムからも参照されている（ここでは、ユーザプログラム２から参照されている）ため削除されない。

【０１０１】

＜実施の形態１における変形例＞

本実施の形態１では、udta atom中に参照回数を示すフィールドを設けたが、本発明はそれに限定されるものではない。

【０１０２】

例えば、この参照回数を示す情報をSample description atom中のSample description table内に設けても良い。すなわち、アフレコデータ毎にSample description を作成するという方法である。これにより、アフレコデータ毎にトラックを分割することを前提としなくても、Sample description atomで管理可能であ

るため、管理ファイルのデータ量の節約に繋がる。ただし、QuickTimeでは、一つのトラックに再生時間の重なるサンプルを同時に置くことはできないため、その場合はトラックを分割する必要がある。そうした場合でも、Sample description table内に参照回数を示す情報を持たせることで、より詳細にアフレコデータを管理することが可能になる。

【０１０３】

例えば、１つのアフレコデータの前半と後半とで異なるコーデックが使われており、Sample descriptionが分かれている場合などで、アフレコデータを記録したユーザプログラムを削除する場合、たとえ他のユーザプログラムからこのアフレコデータが参照されていたとしても、その参照範囲が前半部分のみであれば、後半部分のアフレコデータのみを削除して、そのデータが記録されていた領域を開放することができる。

【０１０４】

また、reference management atom内において、参照回数の情報を示すref#counterだけでなく、参照しているユーザプログラムのファイル名の情報を持たせてもよい。これによって、ユーザによってアフレコデータの削除が指示された場合に、該アフレコデータが他のユーザプログラムファイルから参照されているか否かの情報の他に、どのファイルから参照されているかをユーザに提示することが可能になる。

【０１０５】

さらに、各ユーザプログラムからアフレコデータのどの部分が参照されているかを示す参照範囲の情報を持たせることもできる。このような参照範囲情報により、アフレコデータ自体は複数のユーザプログラムから参照されていても要らなくなった部分のみを削除できるため、より有効にＰＲＵ領域を活用することができる。

【０１０６】

〔実施の形態２〕

次に、本発明の実施の形態２について説明する。実施の形態１と異なり、アフレコしたデータ毎にファイルを分けて管理する。尚、本実施の形態２は上述の実

施の形態１と類似するため、相違点に絞って説明する。

【０１０７】

先ず、アフレコデータが２種類あるときのディスク上の配置とファイルの関係を図２５に示す。光ディスク１０６上の物理的な記録では、ＣＵ毎にアフレコーデータブロックおよびオリジナルブロック（すなわちオリジナルオーディオブロックおよびオリジナルビデオブロック）が配置されており、アフレコデータブロックはアフレコデータ＃Ａを記録する領域、アフレコデータ＃Ｂを記録する領域、未割り当て領域からなっている。そして、アフレコデータ＃Ａに係る各データはアフレコＡファイルに、アフレコデータ＃Ｂに係る各データはアフレコＢファイルに、未割り当て領域はダミーファイルにて管理されている。

【０１０８】

＜初期記録時の処理＞

実施の形態２における初期記録時の処理は、実施の形態１と共通であるので、説明を省略する。

【０１０９】

＜アフレコデータ記録時の処理＞

実施の形態２におけるアフレコデータ記録時の処理を説明する。まず、アフレコデータを追加するユーザプログラムのQuickTime管理情報をＲＡＭ１０２に読み込み、オリジナルプログラムを特定してオリジナルプログラムのQuickTime管理情報をＲＡＭ１０２に読み込む。さらに、ファイルシステム管理情報をＲＡＭ１０２に読み込む。ＰＲＵの位置の特定とアフレコデータ記録領域判定は実施の形態１と同じであるので説明を省く。

【０１１０】

次に、アフレコデータをディスク上のＰＲＵ領域に記録し、新規にアフレコ用のファイルを作成してファイルシステム管理情報を更新する。このとき、新規に作成したアフレコデータのサイズに応じて未割り当て領域で構成されるダミーファイルのサイズを小さくし、ファイルシステム管理情報を更新する。

【０１１１】

次に、オリジナルプログラムのQuickTime管理情報に関して、アフレコデータ

用のトラックを新規に作成し、ユーザが指定した区間のsample tableを作成し、ref#counterフィールドに‘１’をセットする。また、data reference atomには先程作成したアフレコファイル名をセットする。

【０１１２】

次に、オリジナルプログラムの未割り当てトラックのsample tableのchunk offsetとsizeを、アフレコデータ記録後のＰＲＵの残り領域に対応するよう更新する。

【０１１３】

次に、ユーザプログラム側の管理情報に、先程作成したオリジナルプログラムのアフレコトラックのsample tableをコピーする。

【０１１４】

最後に、ＲＡＭ１０２上のオリジナルプログラムおよびユーザプログラムのQuickTime管理情報、及びファイルシステム管理情報を光ディスク１０６に記録する。

【０１１５】

別の種類のアフレコデータを追加するときの処理も上記と同様である。すなわちアフレコデータの種類の数だけファイルが作成される。

【０１１６】

＜ユーザプログラム作成時の処理＞

実施の形態２では、複数のアフレコデータを持つオリジナルプログラムの中からユーザプログラムが参照しているアフレコデータのトラックを特定する際、ユーザプログラムのアフレコトラックのData reference atomに書かれているファイル名と同じファイル名を参照しているトラックをオリジナルプログラムの中からサーチする。オリジナルプログラムは、ユーザプログラムで参照するオーディオビデオファイルから特定する。それ以外の処理は実施の形態１と共通である。

【０１１７】

＜ユーザプログラム削除時の処理＞

実施の形態２では、削除するユーザプログラムのアフレコデータが、オリジナルプログラムのどのトラックで管理されているかを特定する際、上記ユーザプロ

グラム作成時の処理で説明したのと同様に、data reference atomの参照ファイル名を用いて行う。

【０１１８】

また、ユーザプログラムを削除することによって、ref#counterが０になった場合、このアフレコデータをＰＲＵ領域から削除し、開放した領域を未割り当て領域に返却するが、QuickTime管理情報の書き換えに加え、該アフレコデータのファイルも削除し、未割り当て領域で構成されるダミーファイルのサイズを削除したファイルサイズに応じて大きくし、ファイルシステム管理情報の更新をする。それ以外の処理は実施の形態１と共通である。

【０１１９】

このように、実施の形態１では、アフレコファイルのサイズとアドレスは常に変わらないため、アフレコデータの削除が管理情報の書き換えのみで行うことができ、ファイルシステムの書き換えを必要としないものの、アフレコデータを特定する際に、サンプルのchunk offset値からアフレコファイル中の特定データを探索せねばならなかった。

【０１２０】

これに対し、本実施の形態２によると、アフレコデータ毎にファイルが分かれているため、data reference atomの参照ファイル名から一発でアフレコデータを特定することができ、データの特定がより簡便に、より迅速に行うことができる。ただし、管理情報の操作の他、ファイルシステムの操作も必要になる。

【０１２１】

＜実施の形態２における変形例＞

実施の形態２では、オリジナルプログラムの管理情報内で、アフレコデータの種類毎にトラックを分け、その中で参照回数を管理していたが、ディスク上に１個存在する管理情報ファイルであるインデックスファイル内にて管理する構成であっても良い。この場合、図２６に示すように、ファイル名などのアフレコデータファイルを特定するための情報と参照回数情報を関連つけて管理する。

【０１２２】

尚、上記実施の形態１および２において、用いられる記録媒体は光ディスクと

したが、本発明を適用できる記録媒体は光ディスクに限定されるものではなく、ランダムアクセス可能な記録媒体（例えばハードディスク）であればよい。

【０１２３】

【発明の効果】

本発明のデータ記録方法は、以上のように、記録媒体上に、ＡＶデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能であり、ユーザプログラムとアフレコデータとの参照関係を管理する管理情報を上記記録媒体上に記録する構成である。

【０１２４】

それゆえ、例えば、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータを併せて削除しようとする場合には、削除されるべきアフレコデータは削除されるユーザプログラムのみで参照されるものであることを確認することが必要となるが、このような確認が上記管理情報によって容易に行えるといった効果を奏する。

【０１２５】

また、上記データ記録方法では、上記アフレコデータは、複数のユーザプログラムにおいて参照されるアフレコデータ群を含む領域中の部分領域であり、前記領域はひとつのアフレコファイルとして記録されており、上記管理情報は、オリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコデータ毎に記録される構成とすることができる。

【０１２６】

また、上記データ記録方法では、上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、上記管理情報は、複数のオリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコファイル毎に記録される構成とすることができる。

【０１２７】

それゆえ、アフレコファイルのサイズとアドレスとが常に変わらないような管

理が可能であり、アフレコデータの削除が管理情報の書き換えのみで行うことができるため、ファイルシステムの書き換えを必要としないといった効果を奏する。

【０１２８】

また、上記データ記録方法では、上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、上記管理情報は、オリジナルプログラムに対して１つ存在する管理情報ファイル中にアフレコファイル毎に記録される構成とすることができる。

【０１２９】

それゆえ、アフレコデータ毎にアフレコファイルが分かれているため、アフレコデータを特定する際、データのオフセットなどから検索しなくとも、各アフレコファイルの参照ファイル名から即座に特定できるため、データの特定がより簡便に、より迅速に行うことができるといった効果を奏する。

【０１３０】

また、上記データ記録方法では、上記管理情報は、アフレコデータが複数のユーザプログラムから参照されている場合における参照回数を含む構成とすることができる。

【０１３１】

それゆえ、例えば、あるユーザプログラムを削除するにあたって、上記管理情報に含まれる参照回数を確認することによって、削除されるユーザプログラムで参照されているアフレコデータが、この削除されるユーザプログラムのみで参照されるものであるか否かの判断が容易に行えるといった効果を奏する。

【０１３２】

また、上記データ記録方法では、上記管理情報は、アフレコデータを参照しているユーザプログラム名を含む構成とすることができる。

【０１３３】

それゆえ、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータが他のユーザプログラムファイルから参照されているか否かの情報の他に、どのファイルから参照されているかを

ユーザに提示することができるといった効果を奏する。

【０１３４】

また、上記データ記録方法では、上記管理情報は、ユーザプログラムによって参照されているアフレコデータの参照範囲の情報を含む構成とすることができる。

【０１３５】

それゆえ、各ユーザプログラムからアフレコデータのどの部分が参照されているかを示す参照範囲の情報を持たせることで、アフレコデータ自体は複数のユーザプログラムから参照されていても要らなくなった部分のみを削除できるため、より有効に記録媒体の記録領域を活用することができるといった効果を奏する。

【図面の簡単な説明】

【図１】

本発明の一実施形態を示すものであり、アフレコデータ記録時の処理の流れを示したフローチャートである。

【図２】

本発明に係るデータ記録装置の概略構成を示すブロック図である。

【図３】

図３（ａ）～（ｃ）は、QuickTimeファイルフォーマットにおける管理情報とＡＶストリームとの関係を示す図である。

【図４】

QuickTimeファイルフォーマットにおけるMovie atomの概要を示す図である。

【図５】

QuickTimeファイルフォーマットにおけるTrack atomの概要を示す図である。

【図６】

QuickTimeファイルフォーマットにおけるTrack header atomの構成を示す図である。

【図７】

QuickTimeファイルフォーマットにおけるMedia atomの構成を示す図である。

【図８】

QuickTimeファイルフォーマットにおけるMedia information atomの構成を示す図である。

【図９】

Sample table atomによるデータ管理の例を示す図である。

【図１０】

QuickTimeファイルフォーマットにおけるSample table atomの構成を示す図である。

【図１１】

図１１（ａ）はQuickTimeファイルフォーマットにおけるSample description atomの構成を示す図であり、図１１（ｂ）はSound Sample description Tableの構成を示す図である。

【図１２】

QuickTimeファイルフォーマットにおけるEdit atomの構成を示す図である。

【図１３】

図１３（ａ）～（ｃ）は、Edit atomによる再生範囲指定の例を示す図である。

【図１４】

QuickTimeファイルフォーマットにおけるUser data atomの構成を示す図である。

【図１５】

ＡＶストリームの構成を示す図である。

【図１６】

図１６（ａ），（ｂ）は、ＵＤＦにおける管理情報の関係を示す図である。

【図１７】

本発明の第１の実施例における、QuickTimeファイルフォーマットにおけるUser data atomの構成を示す図である。

【図１８】

本発明の実施の形態１における、記録直後のオリジナルプログラムの管理情報を示す図である。

【図１９】

図１９（ａ），（ｂ）は、本発明の実施の形態１における、ユーザプログラム作成時の未割り当てトラックの変更を示す図である。

【図２０】

本発明の実施の形態１における、ディスク上でのデータ配置とファイルとの関係を示した図である。

【図２１】

本発明の実施の形態１における、ユーザプログラムの編集の例を示した図である。

【図２２】

本発明の実施の形態１における、ユーザプログラム削除の処理の流れを示したフローチャートである。

【図２３】

本発明の実施の形態１における、ユーザプログラムの削除直前の参照関係を示した図である。

【図２４】

本発明の実施の形態１における、ユーザプログラムの削除直後の参照関係を示した図である。

【図２５】

本発明の実施の形態２における、ディスク上でのデータの配置とファイルの関係を示す図である。

【図２６】

本発明の実施の形態２における、ユーザプログラムの参照関係とインデックスファイルとの関係を示した図である。

【図２７】

従来技術におけるディスク上での記録形態を示す図である。

【符号の説明】

１００　　バス

１０１　　ホストＣＰＵ

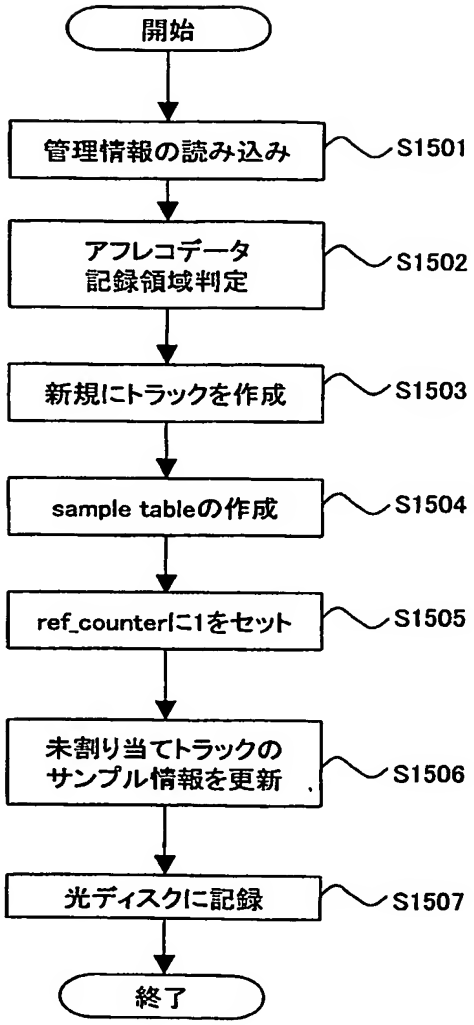
１０２　　ＲＡＭ

１０６　　光ディスク（記録媒体）

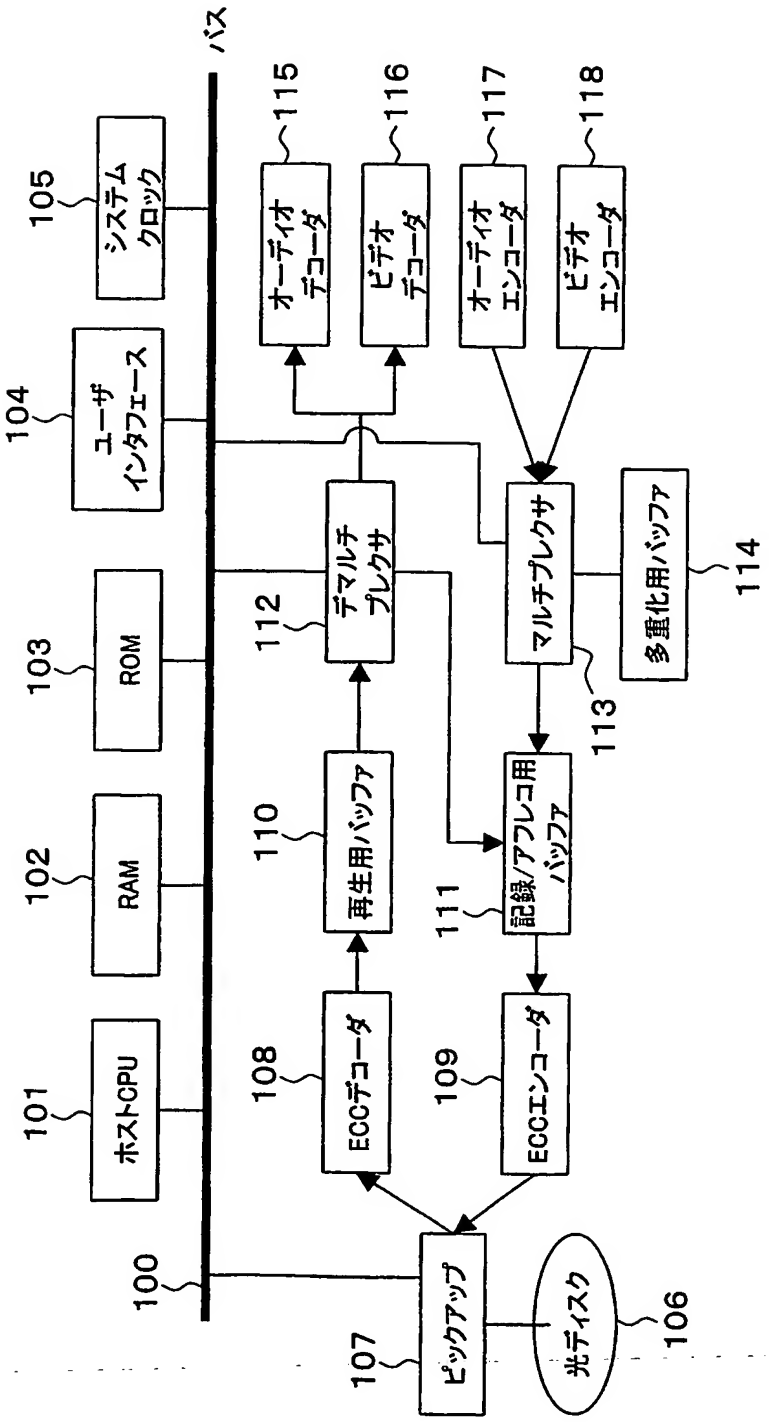
１０７　　ピックアップ

【書類名】　図面

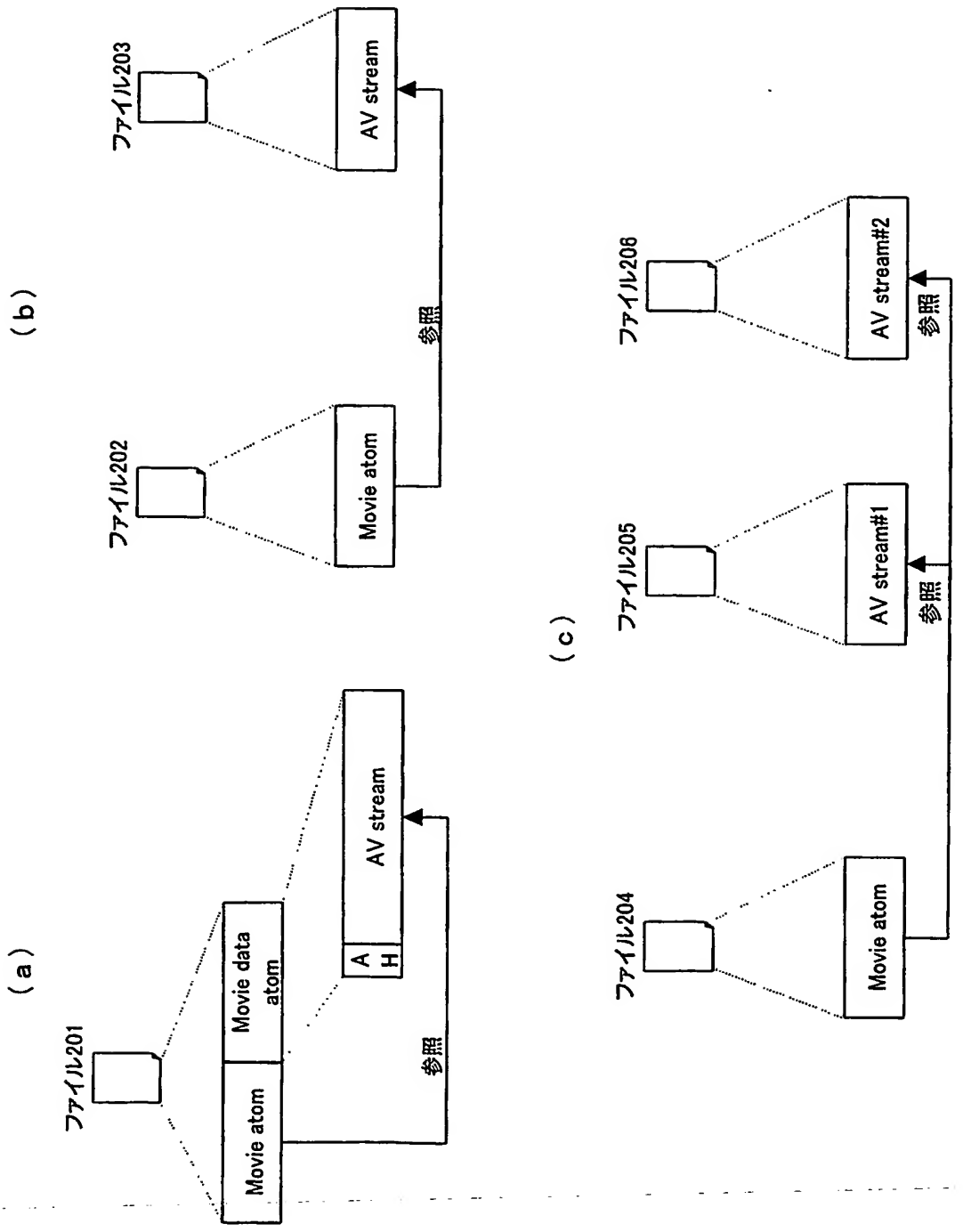
【図１】


開始
管理情報の読み込み
S1501
アフレコデータ
記録領域判定
S1502
新規にトラックを作成
S1503
sample tableの作成
S1504
ref_counterに1をセット
S1505
未割り当てトラックの
サンプル情報を更新
S1506
光ディスクに記録
S1507
終了

【図２】


バス
システム クロック
105
ユーザ インタフェース
104
ROM
103
RAM
102
ホストCPU
101
100
オーディオ デコーダ
115
ビデオ デコーダ
116
オーディオ エンコーダ
117
ビデオ エンコーダ
118
デマルチ プレクサ
112
マルチプレクサ
113
多重化用バッファ
114
再生用バッファ
110
記録/アフレコ用 バッファ
111
ECCデコーダ
108
ECCエンコーダ
109
ピックアップ
107
光ディスク
106

【図３】


(a)
ファイル201
Movie atom
Movie data atom
AV stream
A
H
参照
(b)
ファイル202
Movie atom
ファイル203
AV stream
参照
(c)
ファイル204
Movie atom
ファイル205
AV stream#1
ファイル206
AV stream#2
参照
参照

【図４】

```
Movie atom {
        Atom size
        Type(='moov')
        Movie header atom
        Track atom (video track)
        Track atom (main audio track)
                :
        User data atom
}
```

【図５】

```
Track atom {
        Atom size
        Type('trak')
        Track header  atom
        Edit atom
        Track reference atom
        Media atom
        User data atom
                :
}
```

【図６】

```
Track header atom {
        Atom size
        Type(='tkhd')
        Version
        Flags
        Creation time
        Modification time
        Track ID
        Reserved
        Duration
        Reserved
        Layer
        Alternate group
        Volume
        Reserved
        Matrix structure
        Track width
        Track height
}
```

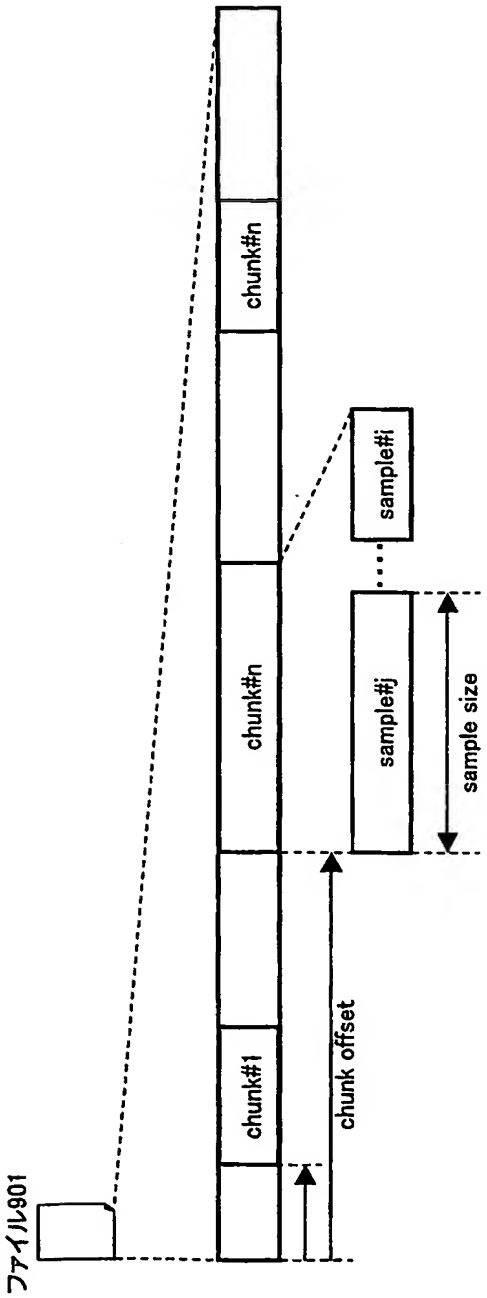
【図７】

```
Media atom {
        Atom size
        Type('mdia')
        Media header  atom
        Handler reference atom
        Media information atom
        User data atom
                :
}
```

【図８】

```
Media information atom {
    Atom size
    Type(='minf')
    {Video or Sound or Base} media information header atom
    Handler reference atom
    Data information atom
    Sample table atom
}
```

【図９】


ファイル801
chunk#1
chunk#n
chunk#n
chunk offset
sample#j
sample size
sample#i

【図１０】

```
Sample table atom {
        Atom size
        Type(='stbl')
        Sample description atom
        Time-to-sample atom
        Sync sample atom
        Sample-to-chunk atom
        Sample size atom
        Chunk offset atom
}
```

【図１１】

(a)

```
Sample description atom {
        Atom size
        Type(='stsd')
        version
        flags
        number-of-entries
        for(i=0; i<number-of-entries; ++){
          Sample Description Table
        }
}
```

(b)

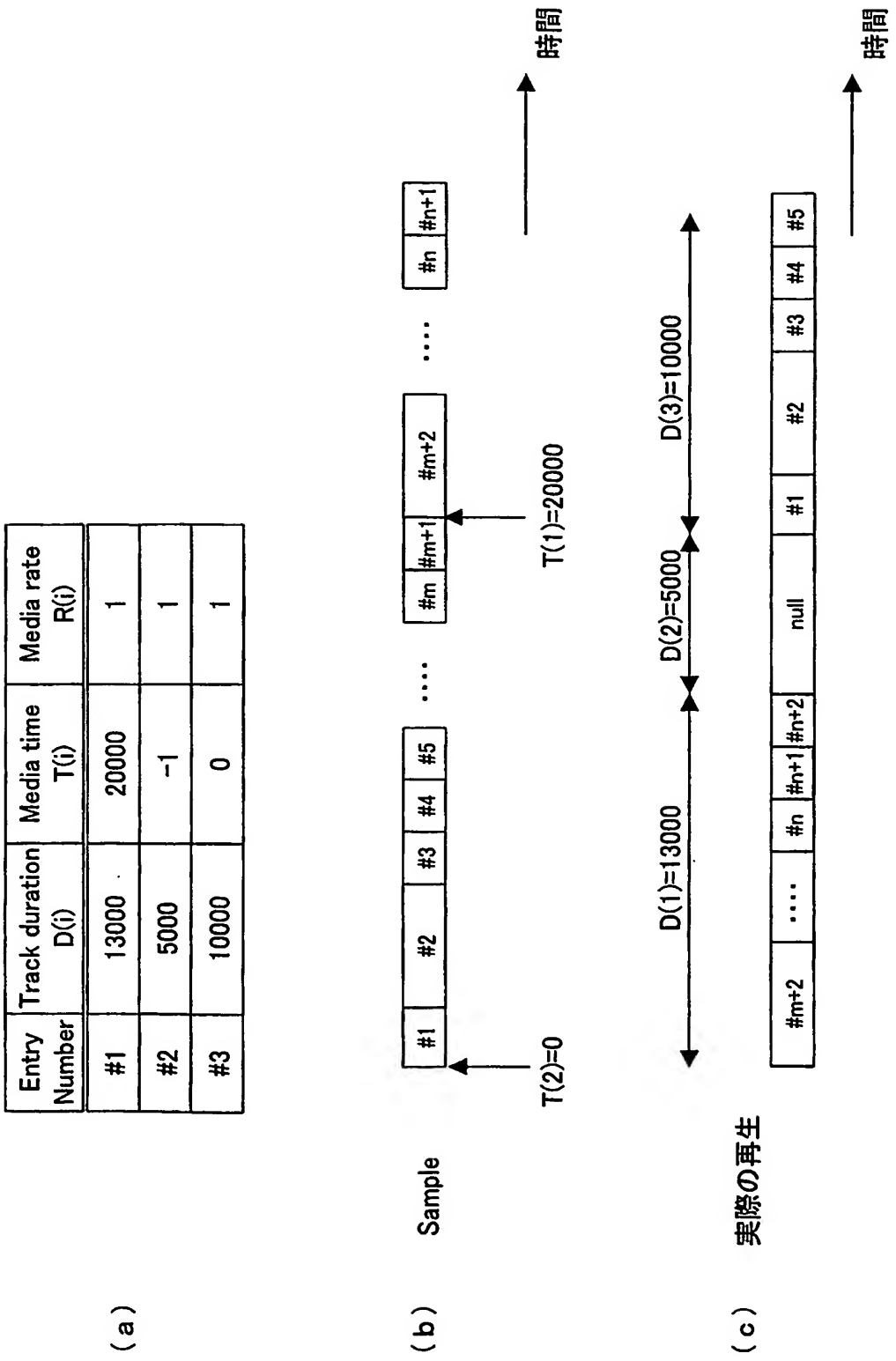
```
Sound Sample description Table {
        size
        data-format
        reserved
        data-reference-index
        //sample description fields for sound media//
        version
        :
}
```

【図１２】

```
Edit atom {
        Atom size
        Type(='edts')
        Edit list atom
}

Edit list atom {
        Atom size
        Type(='elst')
        Versions
        Flags
        Number of entries(=N)
        for (i = 0; i < N; i++){
                Track duration
                Media time
                Media rate
        }
}
```

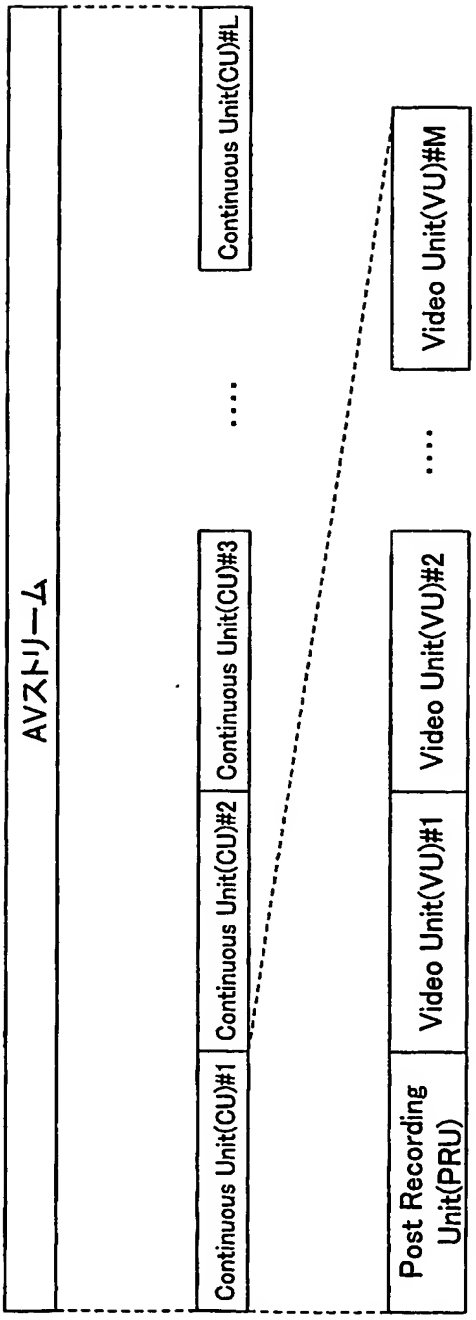
【図１３】


(a)
Entry Number
Track duration D(i)
Media time T(i)
Media rate R(i)
#1
13000
20000
1
#2
5000
-1
1
#3
10000
0
1
(b)
Sample
時間
#1
#2
#3
#4
#5
....
#m
#m+1
#m+2
....
#n
#n+1
T(2)=0
T(1)=20000
(c)
実際の再生
時間
D(1)=13000
D(2)=5000
D(3)=10000
#m+2
....
#n
#n+1
#n+2
null
#1
#2
#3
#4
#5

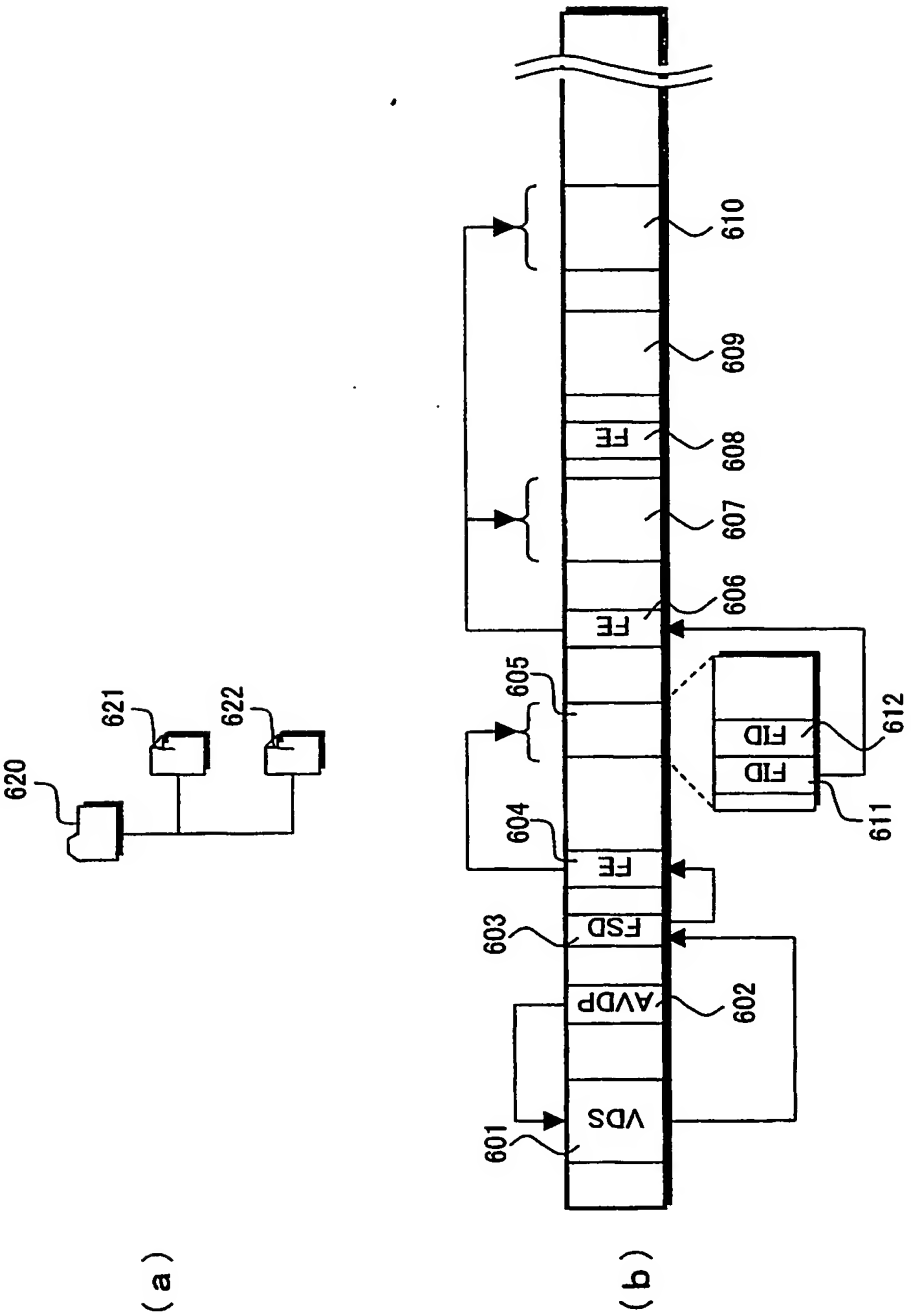
【図１４】

```
User data atom {
        Atom size
        Type(='udta')
        for (i=0;i<N; i++){
                Atom size
                Type
                User data
        }
}
```

【図１５】


AVストリーム
Continuous Unit(CU)#1
Continuous Unit(CU)#2
Continuous Unit(CU)#3
....
Continuous Unit(CU)#L
Post Recording Unit(PRU)
Video Unit(VU)#1
Video Unit(VU)#2
....
Video Unit(VU)#M

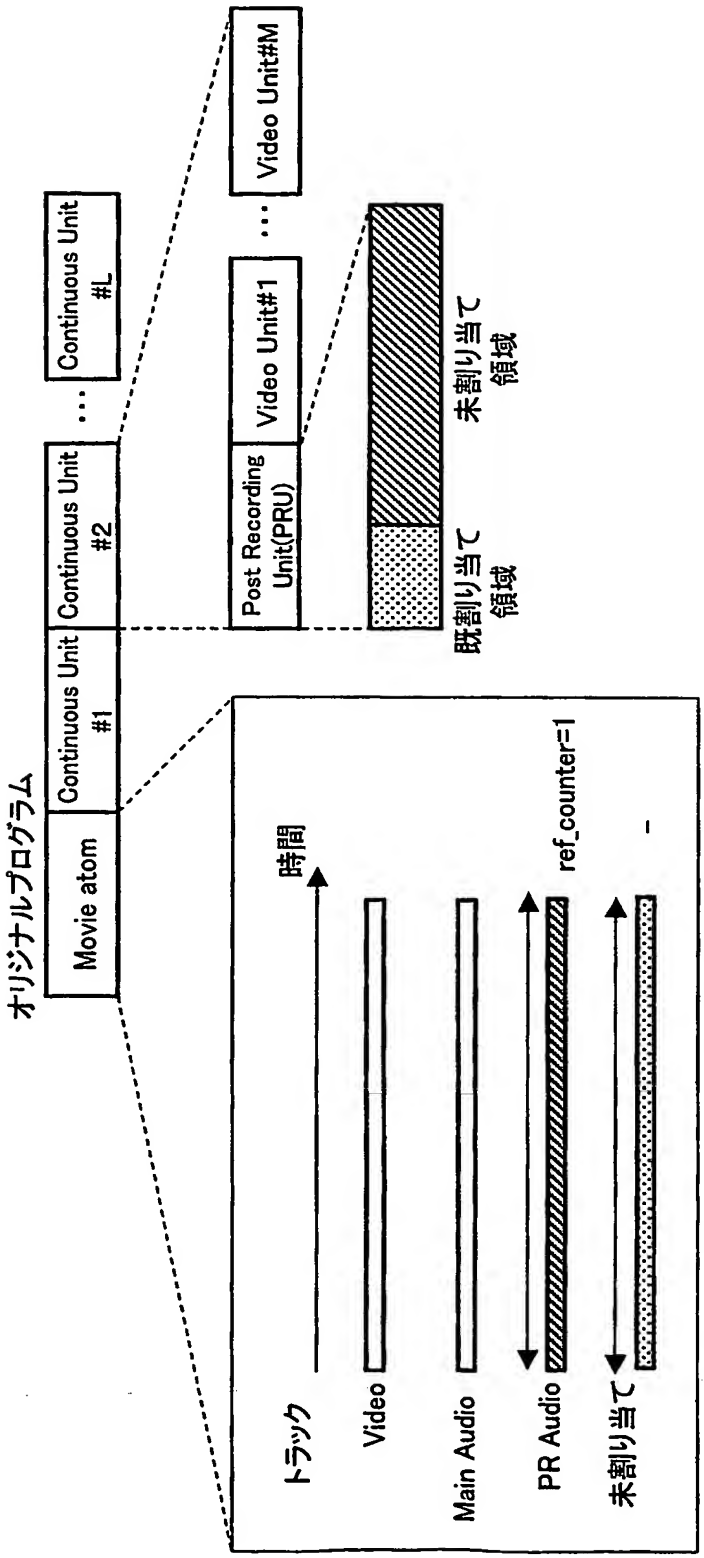
【図１６】


620
621
622
(a)
601
VDS
602
AVDP
603
FSD
604
FE
605
606
FE
607
608
FE
609
610
611
FID
FID
612
(b)

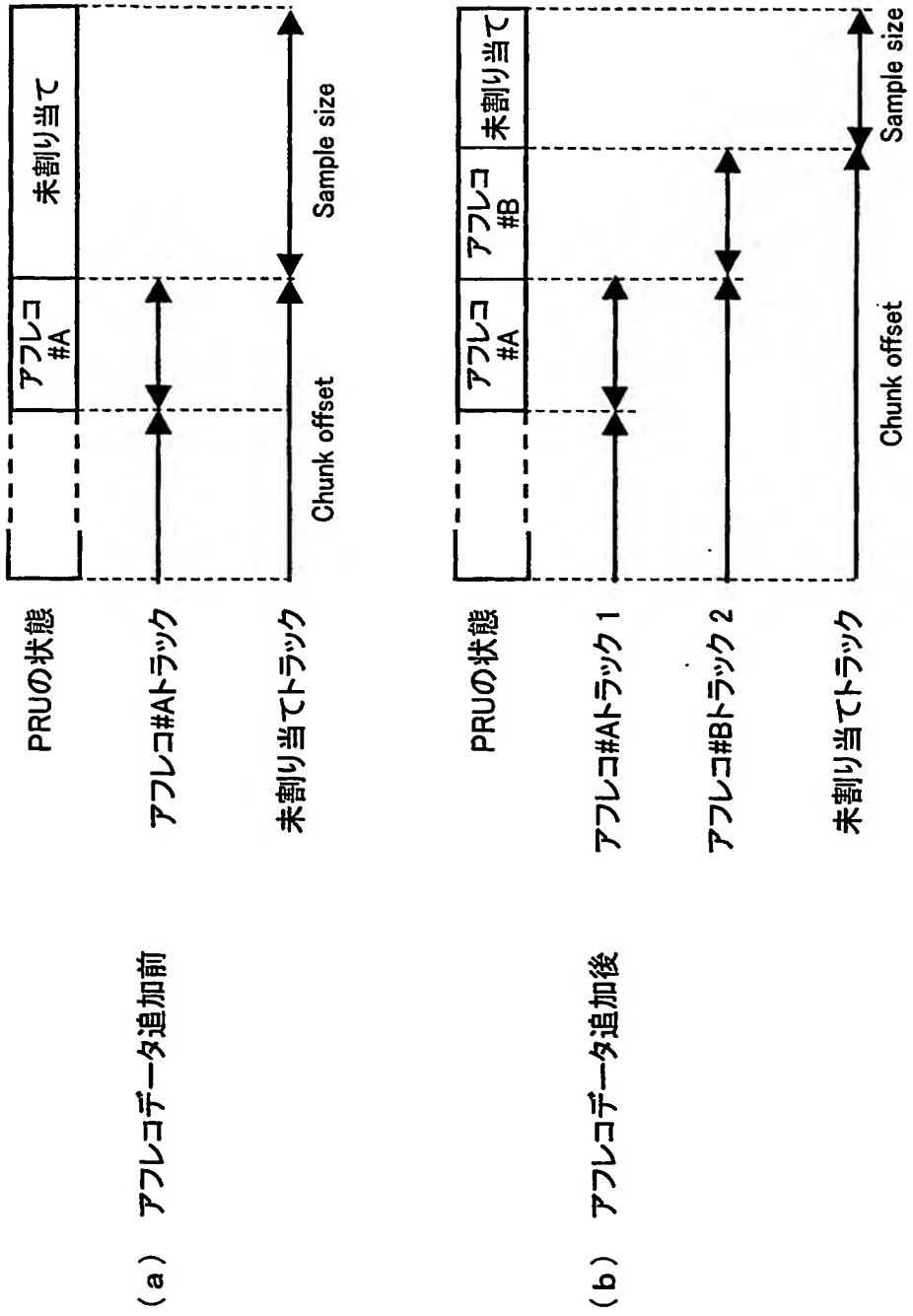
【図１７】

```
User data atom {
        Atom size
        Type(='udta')
        for (i=0;i<N; i++){
                Atom size
                Type('rfmg':reference management)
                ref_counter
        }
}
```

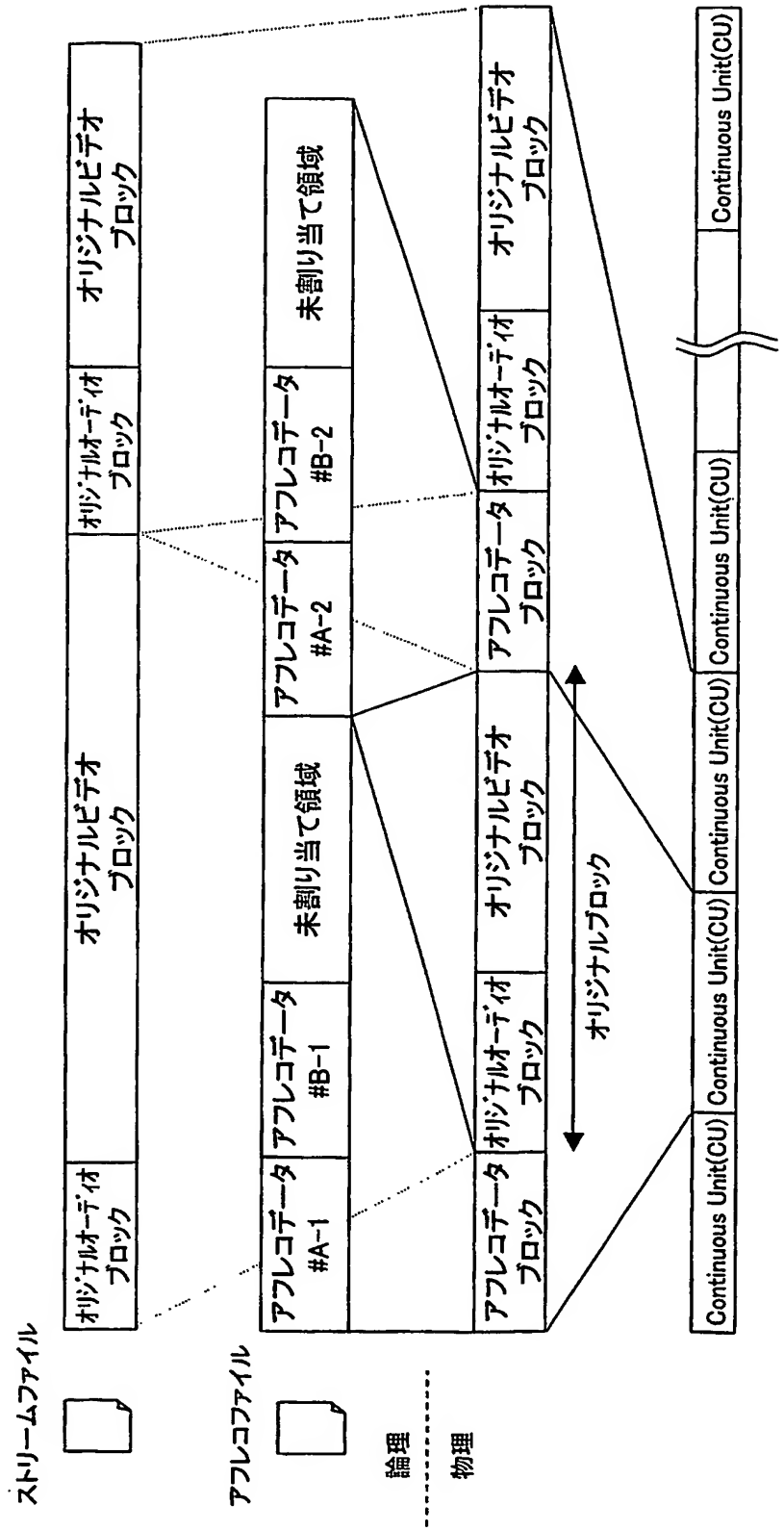
【図１８】


オリジナルプログラム
Movie atom
Continuous Unit #1
Continuous Unit #2
...
Continuous Unit #L
Post Recording Unit(PRU)
Video Unit#1
...
Video Unit#M
既割り当て領域
未割り当て領域
時間
トラック
Video
Main Audio
PR Audio
ref_counter=1
-
未割り当て

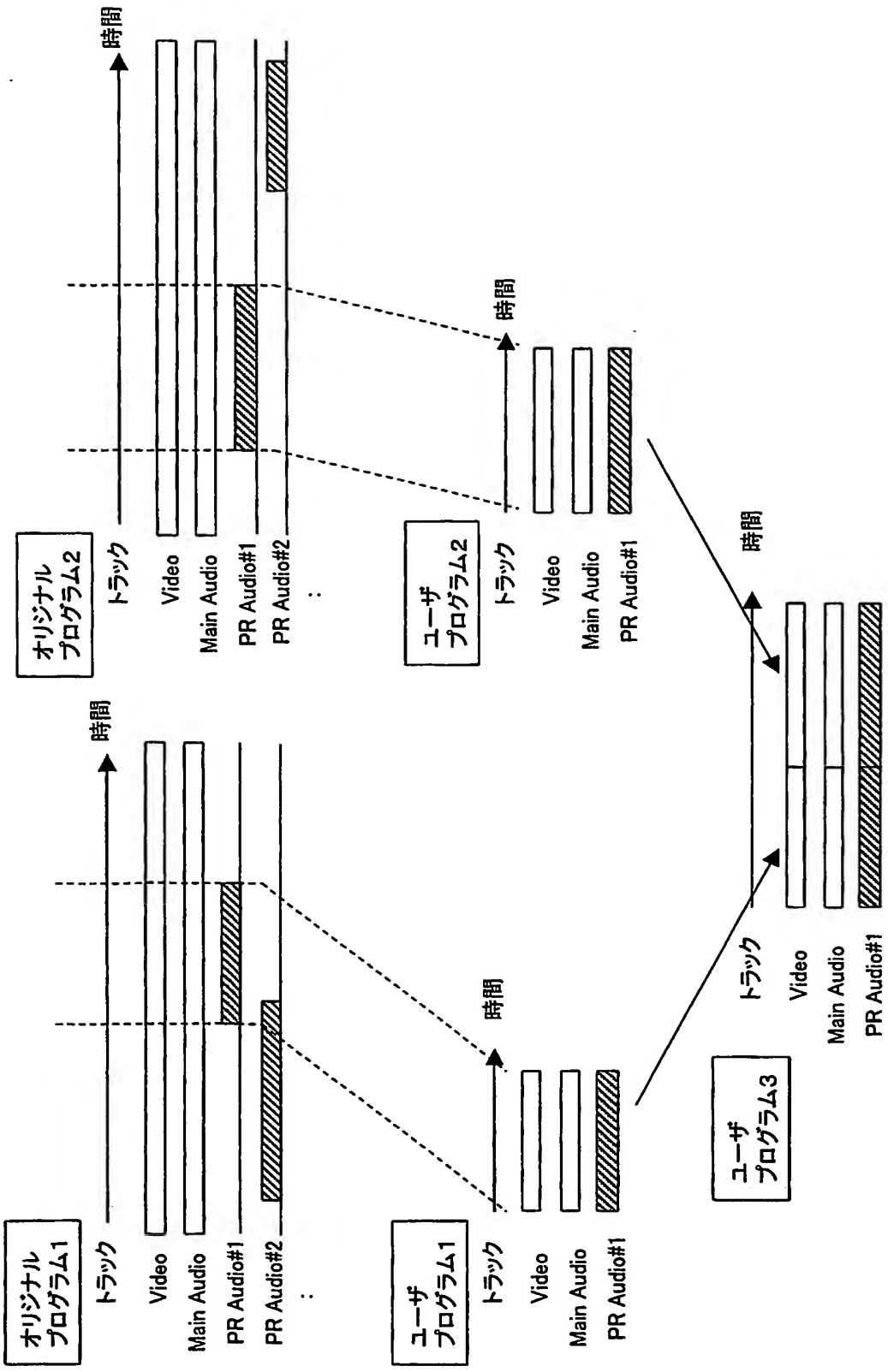
【図１９】


PRUの状態
アフレコ #A
未割り当て
Sample size
Chunk offset
アフレコ#Aトラック
未割り当てトラック
（a）　アフレコデータ追加前
PRUの状態
アフレコ #A
アフレコ #B
未割り当て
Sample size
Chunk offset
アフレコ#Aトラック１
アフレコ#Bトラック２
未割り当てトラック
（b）　アフレコデータ追加後

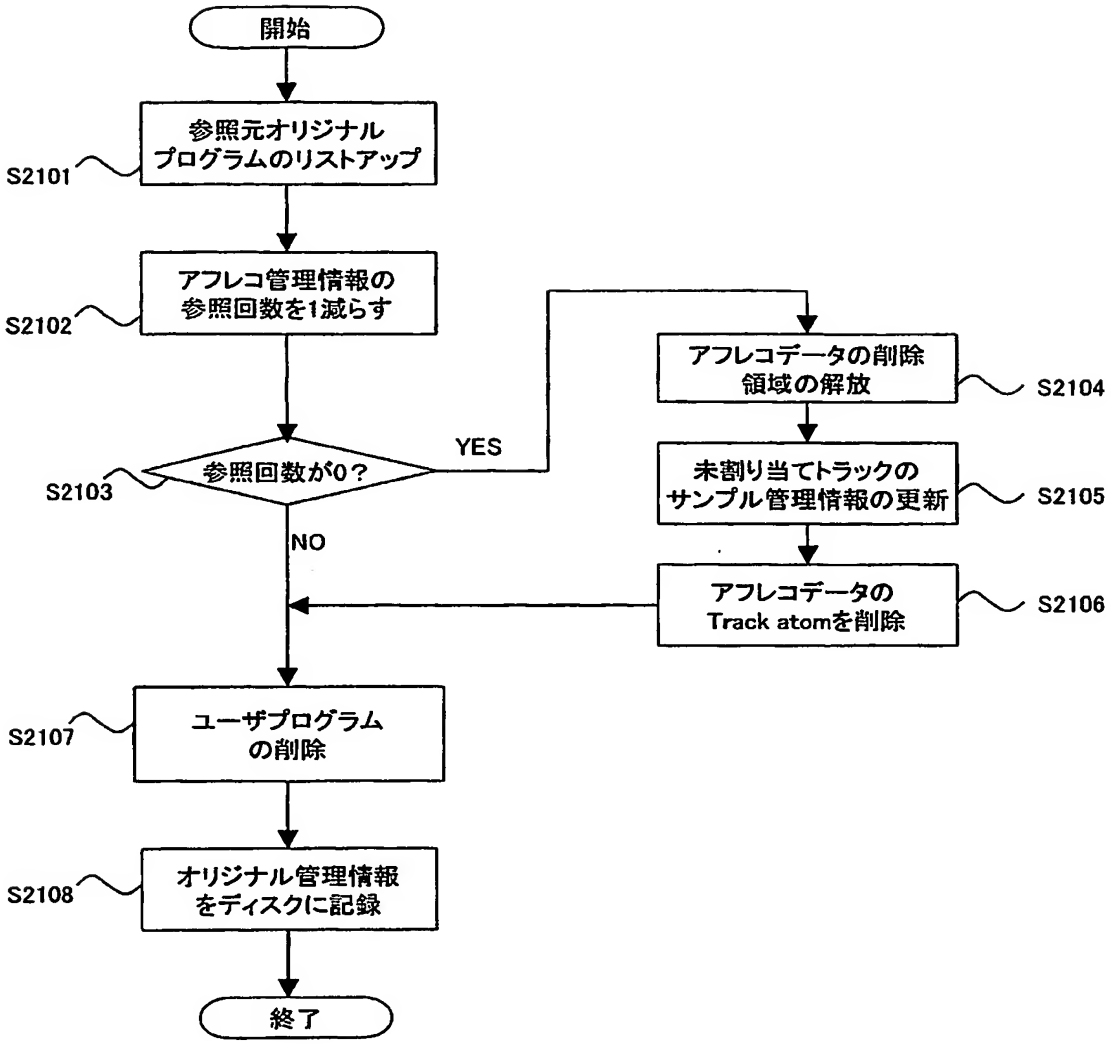
【図２０】


ストリームファイル
アフレコファイル
論理
物理
オリジナルオーディオブロック
オリジナルビデオブロック
アフレコデータ #A-1
アフレコデータ #B-1
未割り当て領域
アフレコデータ #A-2
アフレコデータ #B-2
未割り当て領域
オリジナルオーディオブロック
オリジナルビデオブロック
アフレコデータブロック
オリジナルオーディオブロック
オリジナルビデオブロック
オリジナルブロック
Continuous Unit(CU)
Continuous Unit(CU)
Continuous Unit(CU)
Continuous Unit(CU)

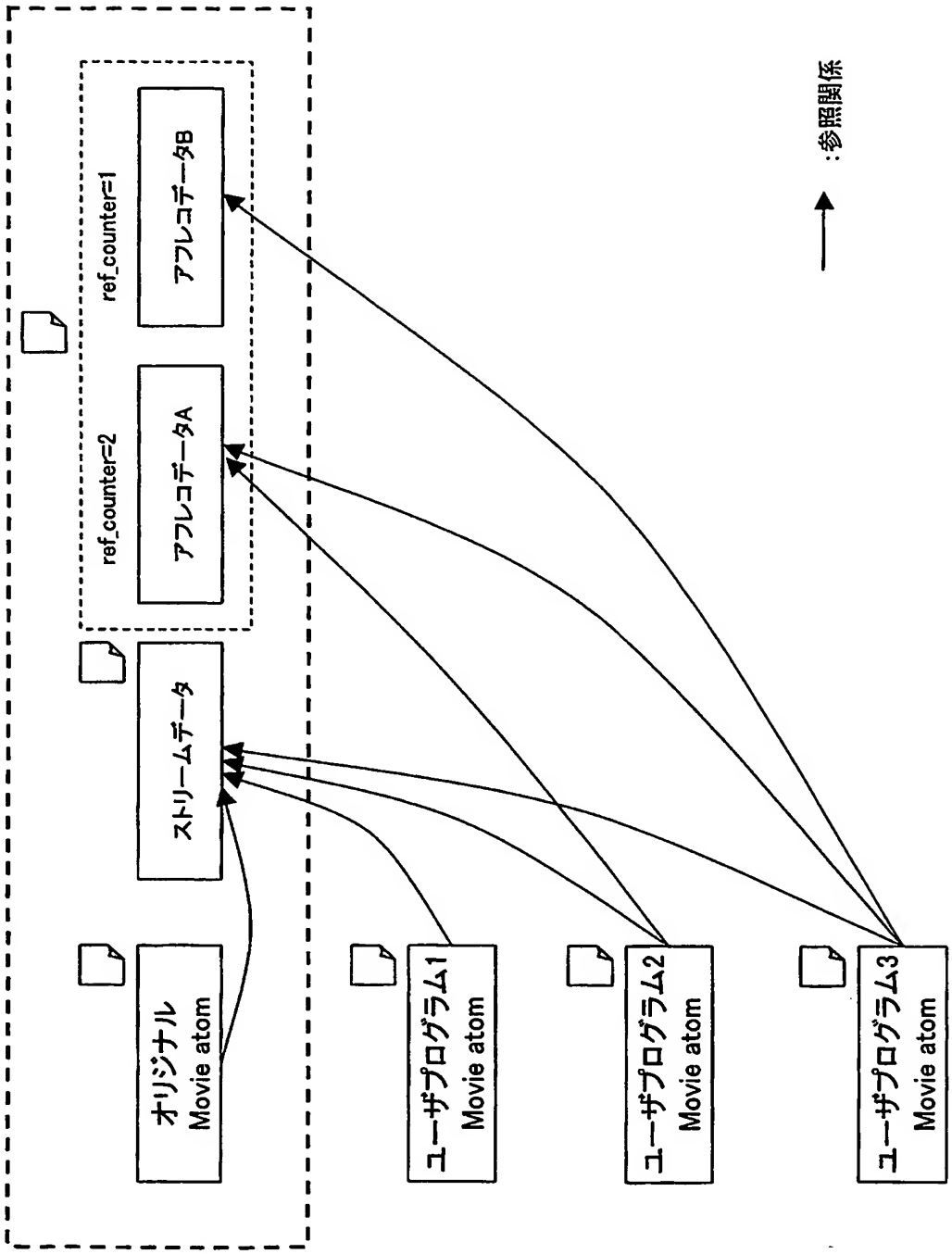
【図２１】


時間
オリジナル
プログラム2
トラック
Video
Main Audio
PR Audio#1
PR Audio#2
ユーザ
プログラム2
ユーザ
プログラム1
ユーザ
プログラム3

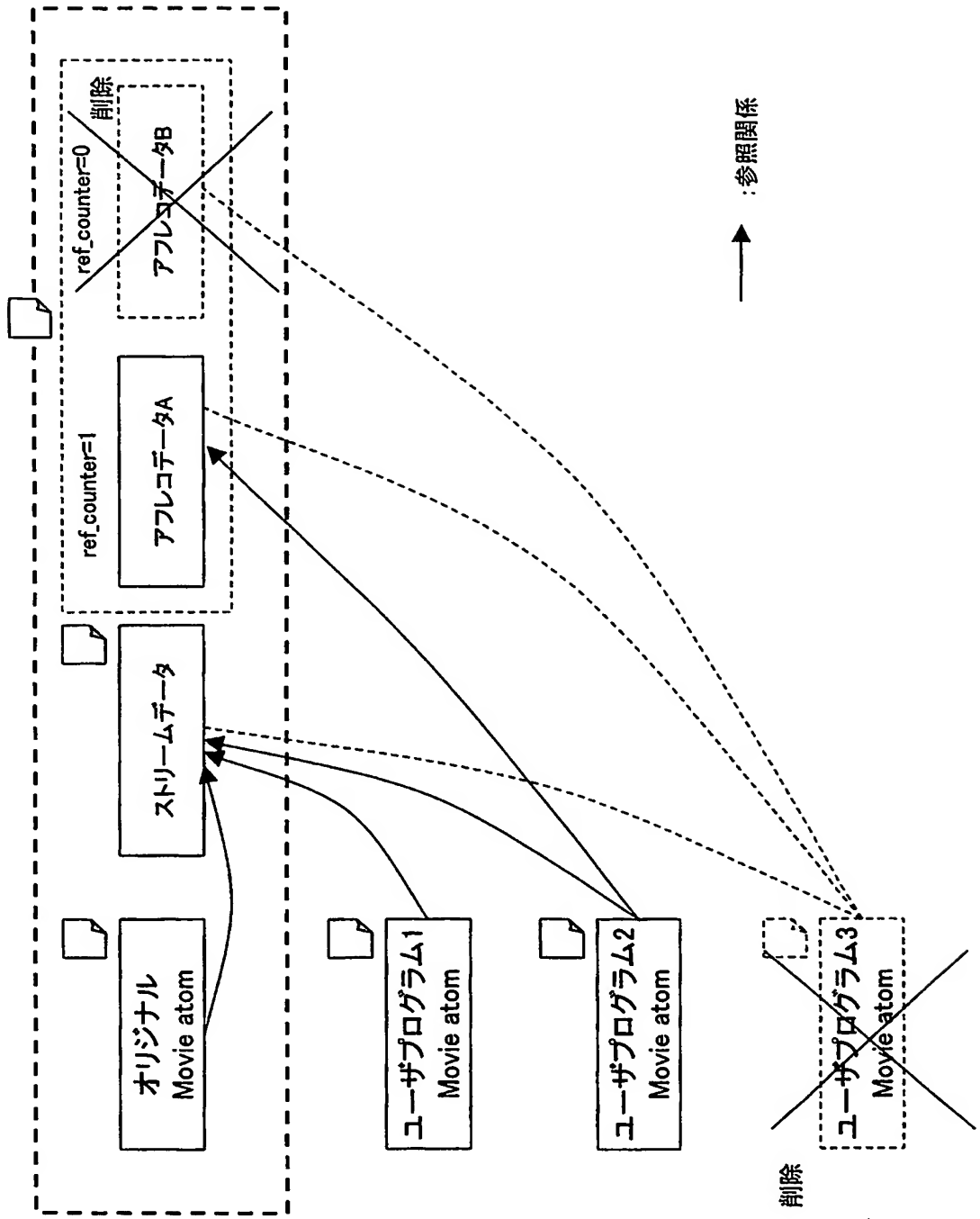
【図２２】


開始
参照元オリジナル
プログラムのリストアップ
S2101
アフレコ管理情報の
参照回数を1減らす
S2102
参照回数が0？
S2103
YES
NO
アフレコデータの削除
領域の解放
S2104
未割り当てトラックの
サンプル管理情報の更新
S2105
アフレコデータの
Track atomを削除
S2106
ユーザプログラム
の削除
S2107
オリジナル管理情報
をディスクに記録
S2108
終了

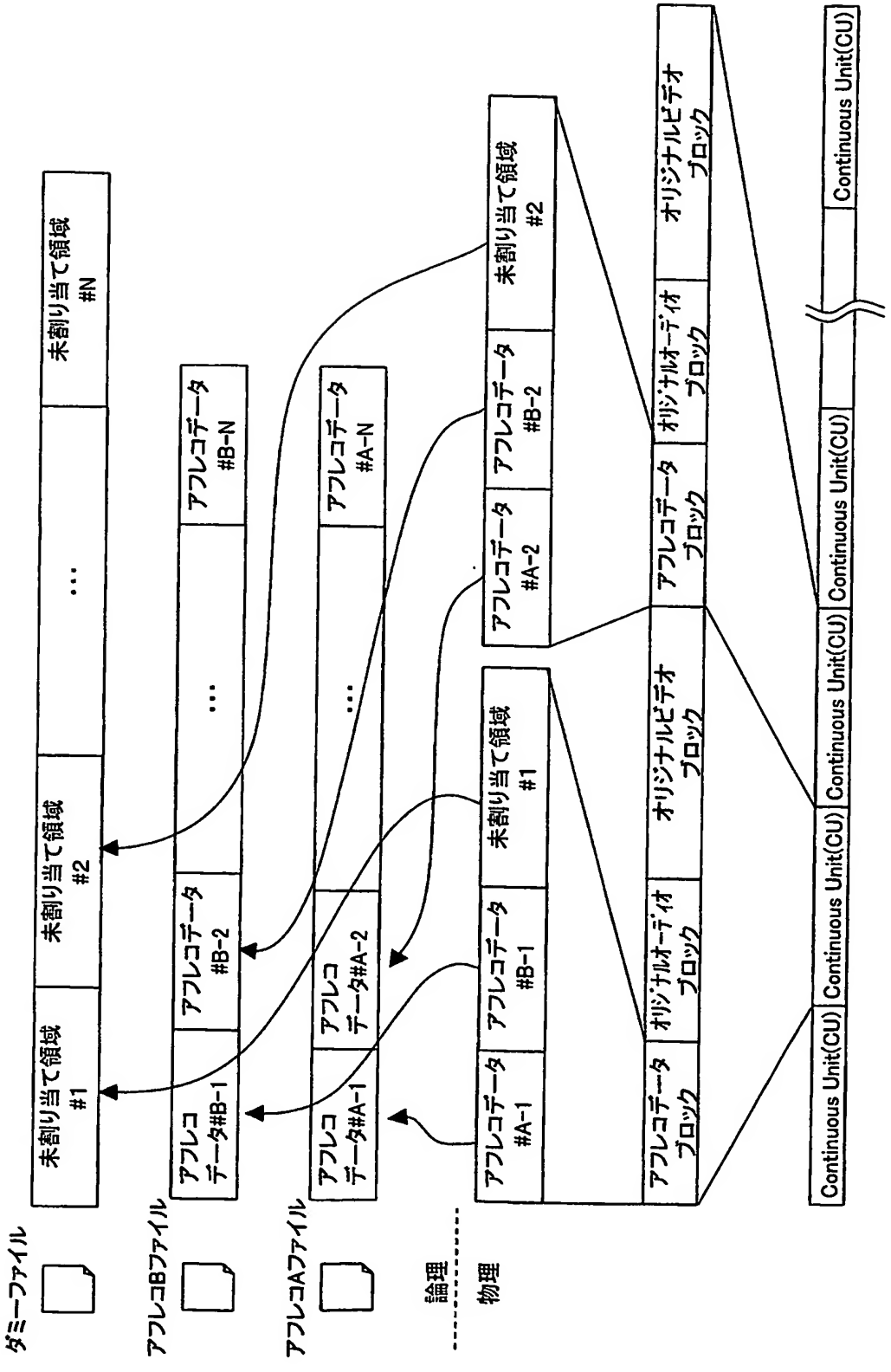
【図２３】


:参照関係
ref_counter=1
ref_counter=2
アフレコデータB
アフレコデータA
ストリームデータ
オリジナル
Movie atom
ユーザプログラム1
Movie atom
ユーザプログラム2
Movie atom
ユーザプログラム3
Movie atom

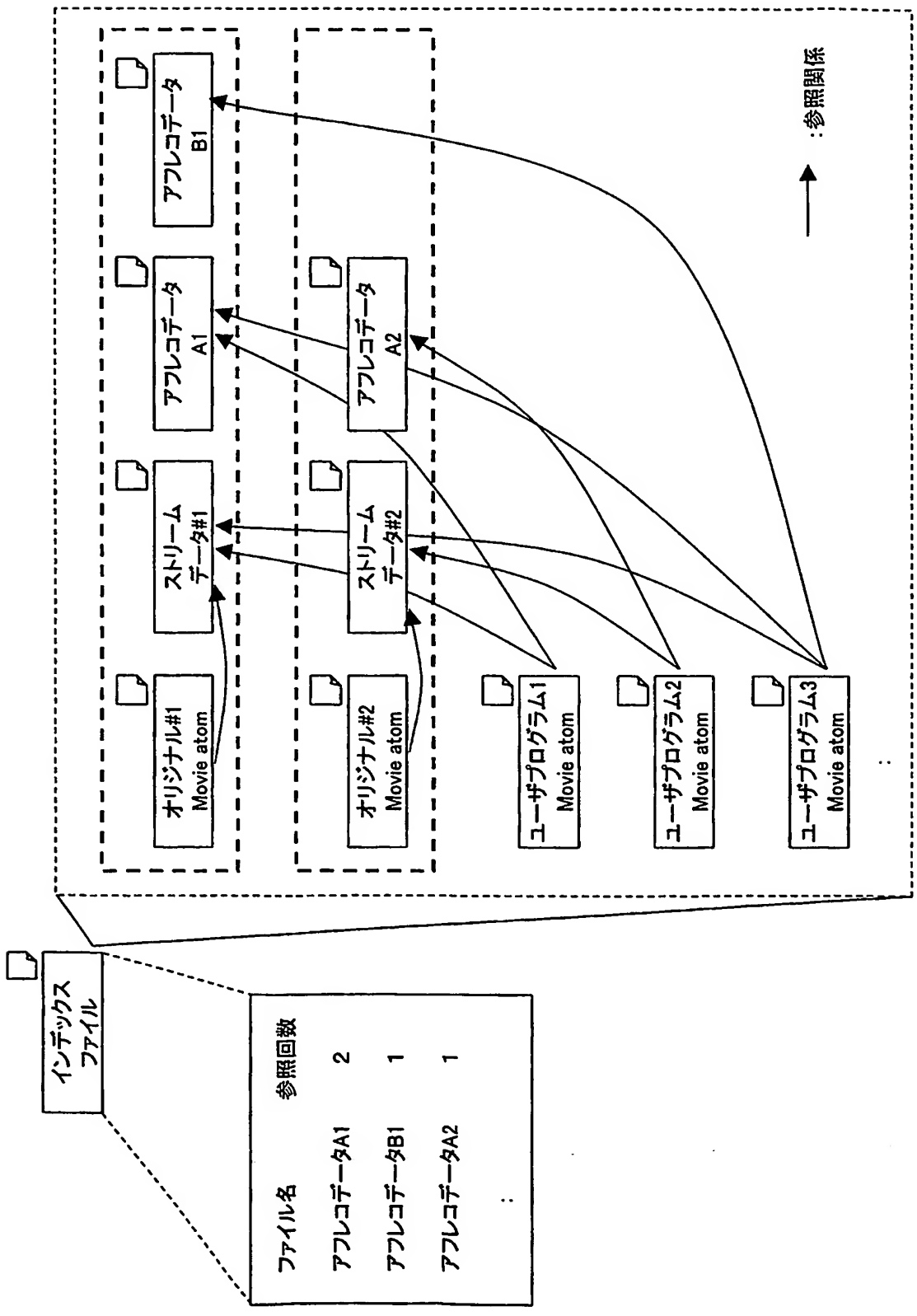
【図２４】


:参照関係
削除
ref_counter=0
アフレコデータB
ref_counter=1
アフレコデータA
ストリームデータ
オリジナル
Movie atom
ユーザプログラム1
Movie atom
ユーザプログラム2
Movie atom
ユーザプログラム3
Movie atom
削除

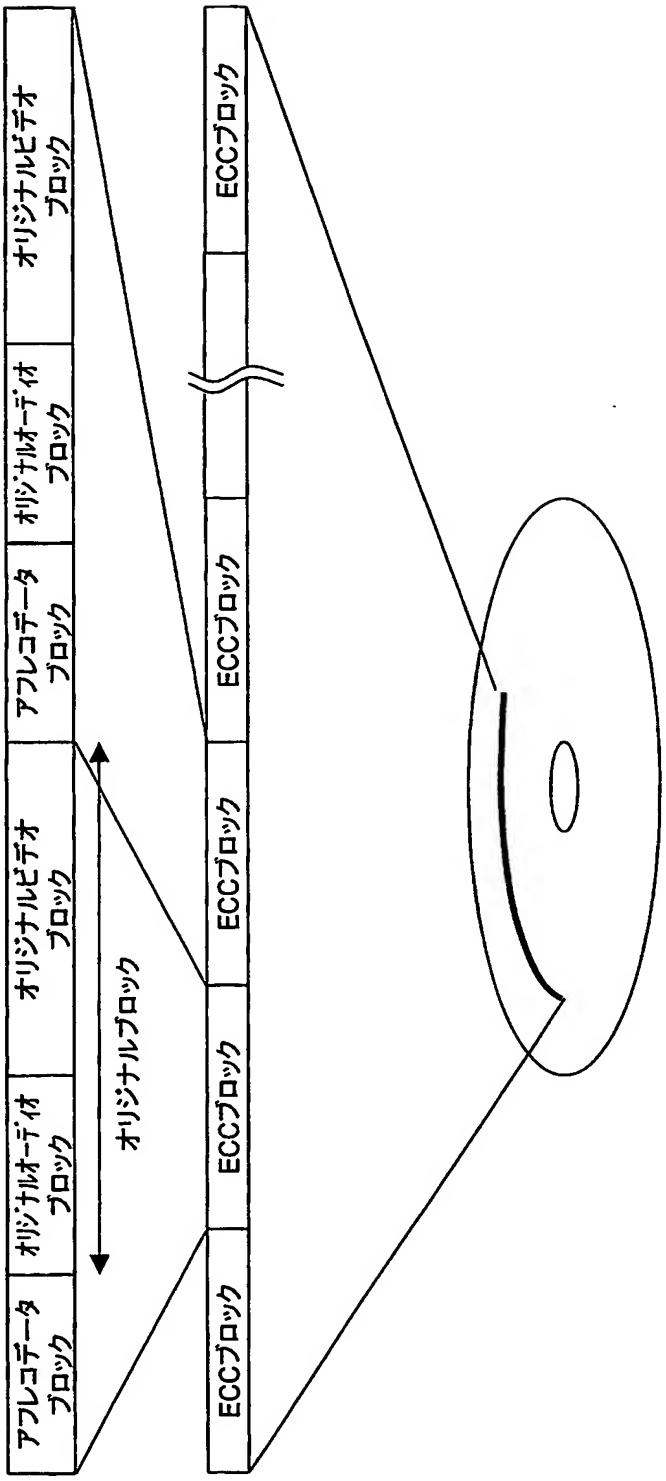
【図２５】


ダミーファイル
アフレコBファイル
アフレコAファイル
論理
物理
未割り当て領域 #1
未割り当て領域 #2
未割り当て領域 #N
アフレコデータ #B-1
アフレコデータ #B-2
アフレコデータ #B-N
アフレコデータ #A-1
アフレコデータ #A-2
アフレコデータ #A-N
未割り当て領域 #1
未割り当て領域 #2
アフレコデータ #A-1
アフレコデータ #A-2
アフレコデータ #B-1
アフレコデータ #B-2
オリジナルオーディオブロック
オリジナルビデオブロック
アフレコデータブロック
Continuous Unit(CU)

【図２６】


インデックス
ファイル
ファイル名　参照回数
アフレコデータA1　2
アフレコデータB1　1
アフレコデータA2　1
オリジナル#1
Movie atom
ストリーム
データ#1
アフレコデータ
A1
アフレコデータ
B1
オリジナル#2
Movie atom
ストリーム
データ#2
アフレコデータ
A2
ユーザプログラム1
Movie atom
ユーザプログラム2
Movie atom
ユーザプログラム3
Movie atom
:参照関係

【図２７】


オリジナルビデオブロック
オリジナルオーディオブロック
アフレコデータブロック
オリジナルビデオブロック
オリジナルオーディオブロック
アフレコデータブロック
オリジナルブロック
ECCブロック
ECCブロック
ECCブロック
ECCブロック
ECCブロック

【書類名】　要約書

【要約】

【課題】　ユーザプログラムの削除時、該ユーザプログラムにおいて参照していたアフレコデータが削除可能であるか否かの判断を容易にする。

【解決手段】　オリジナルプログラムを外部参照したユーザプログラムを作成し、それにアフレコデータを記録するにあたって、アフレコデータのユーザプログラムからの参照回数を示すref#counterを設定する（Ｓ１５０５）。設定されたref#counterは、作成されたユーザプログラムと共に光ディスクに記録される（Ｓ１５０７）。

【選択図】　図１

特願２００３－１６２９２８

出願人履歴情報

識別番号　［０００００５０４９］

１．変更年月日　１９９０年　８月２９日
［変更理由］　新規登録
住　所　大阪府大阪市阿倍野区長池町２２番２２号
氏　名　シャープ株式会社